Repository Codes

Repository: ai_semantic_search

File: README.md

```
# BrainQuery - AI Semantic Search
Do you have difficulty finding relevant research papers? BrainQuery to the rescue! It employs AI Semantic Search to find hidden meaning behind searches and uses that to search through the abstracts of papers from the arXiv database - a large open-access archive of scientific material. (and yes, it has dark mode too!)

It even includes multi language support using OpenAI translation, which makes searches possible in any language.
## Technologies Used:

### Frontend:
- ReactJS
- Tailwind CSS

### Backend:
- Flask API
- OpenAI API (for Text Embeddings and Translation)
- Pinecone (Vector Database)

## Screenshots:
![Dark page](/images/dark.jpg?raw=true "Dark Page")

### Example Query: "minimum resource quantum computation increase"
![Query1](/images/query1.png?raw=true "Query1")

### Example Query (German): "I-V-Eigenschaften von MgB2" (translation: I-V properties of MgB2)
![Query2](/images/query2.png?raw=true "Query2")

### Example Query (Bengali): "??? ??????? ?????????? ?????????? ??????" (translation: min resource quantum computation increase)
![Query3](/images/query3.png?raw=true "Query3")

## How It Works:
The dataset consists of the arXiv Dataset (https://www.kaggle.com/datasets/Cornell-University/arxiv), of which, the first 15000 entries have been used for this project (due to Pinecone Restrictions). The embeddings for the abstract of each article were created (using OpenAI's ada-002 model), which has a dimensionality of 1536. These embeddings were then indexed in the Pincone Vector Database.
For Queries, this project uses OpenAI's translation (text completion via text-davinci-003 model) and embeddings (ada-002 model) to first translate given query into English and then embed it. A query is then made to the Pinecone database, which computes the top 5 items with consine similarity and returns the results as per ranking (most similar first), which is then displayed using the React frontend.
```

File: app.py

```
from dotenv import load_dotenv
from flask import Flask, request
import os
import openai
import pinecone
import json

app = Flask(__name__)

final = {}

# Loading environment variables
load_dotenv()
OPENAI_KEY = os.getenv("OPENAI_KEY")
PINECONE_KEY = os.getenv("PINECONE_KEY")

openai.api_key = OPENAI_KEY

# Selecting ada-002 model for text embeddings
MODEL = "text-embedding-ada-002"

# Initialize pinecone connection
pinecone.init(
api_key=PINECONE_KEY,
environment="us-west1-gcp"  # find next to api key in   console
)

# Select index name and connect to index
index_name = 'semantic-search'
index = pinecone.Index(index_name)


@app.route('/predictor', methods=["POST"])
def predict():
query = request.get_json()
# Translation API call
response = openai.Completion.create(
model="text-davinci-003",
prompt="Translate this to English (if already in english, just return input text): " +
query,
temperature=0.3,
max_tokens=15,
top_p=1.0,
frequency_penalty=0.0,
presence_penalty=0.0
)

# Created embeddings for translated text

xq            =            openai.Embedding.create(input=response['choices'][0]['text'],
engine=MODEL)['data'][0]['embedding']
res = index.query([xq], top_k=5, include_metadata=True)
```

```
for i in range(len(res['matches'])):
final[i] = res['matches'][i]['id']

json_obj = json.dumps(final)

return json_obj
```

File: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<meta name="theme-color" content="#000000" />
<meta name="description" content="Web site created using create-react-app"/>
<link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<title>BrainQuery</title>
</head>
<body>
<div id="root"></div>
</body>
</html>
```

File: robots.txt

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:
```

Repository: ClearSkies

File: README.md

```
# ClearSkies
## Clear Skies: Predicting and Preventing Air Pollution

Air Pollution has been a long standing problem - specially in some urban cities of India. ClearSkies is a web app which aims to serve as a hub for getting education info, news, and serve predictions about Air Quality using a machine learning model

## Technologies Used:

### Frontend:
- ReactJS
- CSS

### Backend: (Planned, for serving Predictions and News)
- Django API
- Integrating News API for serving News regarding Air Pollution

### Model Training:
- XGBoost

# Screenshots

### Home Page:
![Home page](/assets/home.png?raw=true "Home Page")

### Educational Page (In progress):
![Edu page](/assets/placeholder_edu.png?raw=true "Edu Page")

### News Page (In progress):
![News page](/assets/placeholder_news.png?raw=true "Home Page")


### Predictions Page (In progress):
![Predictions page](/assets/placeholder_form.png?raw=true "Home Page")
```

File: model.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 1,
"metadata": {},
"outputs": [],
"source": [
```

```
"import numpy as np\n",
"import pandas as pd"
]
},
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [],
"source": [
"df = pd.read_csv(\"./dataset/city_day.csv\")"
]
},
{
"cell_type": "code",
"execution_count": 3,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>Date</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
```

```
"      <th>AQI_Bucket</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-01</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>NaN</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-02</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>NaN</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-03</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>NaN</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
```

```
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-04</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>NaN</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-05</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>NaN</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
```

```
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>95</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-06</td>\n",
"      <td>120.96</td>\n",
"      <td>NaN</td>\n",
"      <td>4.23</td>\n",
"      <td>20.86</td>\n",
"      <td>25.02</td>\n",
"      <td>NaN</td>\n",
"      <td>4.23</td>\n",
"      <td>35.54</td>\n",
"      <td>35.66</td>\n",
"      <td>6.05</td>\n",
"      <td>15.85</td>\n",
"      <td>1.68</td>\n",
"      <td>303.0</td>\n",
"      <td>Very Poor</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>96</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-07</td>\n",
"      <td>138.63</td>\n",
"      <td>NaN</td>\n",
"      <td>5.65</td>\n",
"      <td>25.47</td>\n",
"      <td>30.67</td>\n",
"      <td>NaN</td>\n",
"      <td>5.65</td>\n",
"      <td>87.43</td>\n",
"      <td>29.77</td>\n",
"      <td>11.25</td>\n",
"      <td>24.67</td>\n",
"      <td>3.12</td>\n",
"      <td>314.0</td>\n",
"      <td>Very Poor</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>97</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-08</td>\n",
"      <td>144.36</td>\n",
"      <td>NaN</td>\n",
```

```
"      <td>15.82</td>\n",
"      <td>30.69</td>\n",
"      <td>46.10</td>\n",
"      <td>NaN</td>\n",
"      <td>15.82</td>\n",
"      <td>66.39</td>\n",
"      <td>27.34</td>\n",
"      <td>17.22</td>\n",
"      <td>49.56</td>\n",
"      <td>7.71</td>\n",
"      <td>378.0</td>\n",
"      <td>Very Poor</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>98</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-09</td>\n",
"      <td>85.03</td>\n",
"      <td>NaN</td>\n",
"      <td>3.30</td>\n",
"      <td>26.70</td>\n",
"      <td>27.41</td>\n",
"      <td>NaN</td>\n",
"      <td>3.30</td>\n",
"      <td>59.16</td>\n",
"      <td>35.89</td>\n",
"      <td>7.58</td>\n",
"      <td>29.19</td>\n",
"      <td>4.08</td>\n",
"      <td>415.0</td>\n",
"      <td>Severe</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>99</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-10</td>\n",
"      <td>134.60</td>\n",
"      <td>NaN</td>\n",
"      <td>5.90</td>\n",
"      <td>26.27</td>\n",
"      <td>28.70</td>\n",
"      <td>NaN</td>\n",
"      <td>5.90</td>\n",
"      <td>34.53</td>\n",
"      <td>7.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>100 rows × 16 columns</p>\n",
```

```
"</div>"
],
"text/plain": [
"         City        Date   PM2.5  PM10     NO    NO2    NOx  NH3     CO  \\\n",
"0   Ahmedabad  2015-01-01     NaN   NaN   0.92  18.22  17.15  NaN   0.92   \n",
"1   Ahmedabad  2015-01-02     NaN   NaN   0.97  15.69  16.46  NaN   0.97   \n",
"2   Ahmedabad  2015-01-03     NaN   NaN  17.40  19.30  29.70  NaN  17.40   \n",
"3   Ahmedabad  2015-01-04     NaN   NaN   1.70  18.48  17.97  NaN   1.70   \n",
"4   Ahmedabad  2015-01-05     NaN   NaN  22.10  21.42  37.76  NaN  22.10   \n",
"..        ...         ...     ...   ...    ...    ...    ...  ...    ...   \n",
"95  Ahmedabad  2015-04-06  120.96   NaN   4.23  20.86  25.02  NaN   4.23   \n",
"96  Ahmedabad  2015-04-07  138.63   NaN   5.65  25.47  30.67  NaN   5.65   \n",
"97  Ahmedabad  2015-04-08  144.36   NaN  15.82  30.69  46.10  NaN  15.82   \n",
"98  Ahmedabad  2015-04-09   85.03   NaN   3.30  26.70  27.41  NaN   3.30   \n",
"99  Ahmedabad  2015-04-10  134.60   NaN   5.90  26.27  28.70  NaN   5.90   \n",
"\n",
"      SO2      O3  Benzene  Toluene  Xylene    AQI AQI_Bucket  \n",
"0   27.64  133.36     0.00     0.02    0.00    NaN        NaN  \n",
"1   24.55   34.06     3.68     5.50    3.77    NaN        NaN  \n",
"2   29.07   30.70     6.80    16.40    2.25    NaN        NaN  \n",
"3   18.59   36.08     4.43    10.14    1.00    NaN        NaN  \n",
"4   39.33   39.31     7.01    18.89    2.78    NaN        NaN  \n",
"..    ...     ...      ...      ...     ...    ...        ...  \n",
"95  35.54   35.66     6.05    15.85    1.68  303.0  Very Poor  \n",
"96  87.43   29.77    11.25    24.67    3.12  314.0  Very Poor  \n",
"97  66.39   27.34    17.22    49.56    7.71  378.0  Very Poor  \n",
"98  59.16   35.89     7.58    29.19    4.08  415.0     Severe  \n",
"99  34.53    7.00     0.00     0.00    0.00    NaN        NaN  \n",
"\n",
"[100 rows x 16 columns]"
]
},
"execution_count": 3,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head(100)"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"(29531, 16)"
]
},
"execution_count": 4,
"metadata": {},
```

```
"output_type": "execute_result"
}
],
"source": [
"df.shape"
]
},
{
"cell_type": "code",
"execution_count": 5,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>24933.000000</td>\n",
"      <td>18391.000000</td>\n",
"      <td>25949.000000</td>\n",
```

```
"      <td>25946.000000</td>\n",
"      <td>25346.000000</td>\n",
"      <td>19203.000000</td>\n",
"      <td>27472.000000</td>\n",
"      <td>25677.000000</td>\n",
"      <td>25509.000000</td>\n",
"      <td>23908.000000</td>\n",
"      <td>21490.000000</td>\n",
"      <td>11422.000000</td>\n",
"      <td>24850.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>mean</th>\n",
"      <td>67.450578</td>\n",
"      <td>118.127103</td>\n",
"      <td>17.574730</td>\n",
"      <td>28.560659</td>\n",
"      <td>32.309123</td>\n",
"      <td>23.483476</td>\n",
"      <td>2.248598</td>\n",
"      <td>14.531977</td>\n",
"      <td>34.491430</td>\n",
"      <td>3.280840</td>\n",
"      <td>8.700972</td>\n",
"      <td>3.070128</td>\n",
"      <td>166.463581</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>std</th>\n",
"      <td>64.661449</td>\n",
"      <td>90.605110</td>\n",
"      <td>22.785846</td>\n",
"      <td>24.474746</td>\n",
"      <td>31.646011</td>\n",
"      <td>25.684275</td>\n",
"      <td>6.962884</td>\n",
"      <td>18.133775</td>\n",
"      <td>21.694928</td>\n",
"      <td>15.811136</td>\n",
"      <td>19.969164</td>\n",
"      <td>6.323247</td>\n",
"      <td>140.696585</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>min</th>\n",
"      <td>0.040000</td>\n",
"      <td>0.010000</td>\n",
"      <td>0.020000</td>\n",
"      <td>0.010000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.010000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.010000</td>\n",
"      <td>0.010000</td>\n",
```

```
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>13.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>25%</th>\n",
"      <td>28.820000</td>\n",
"      <td>56.255000</td>\n",
"      <td>5.630000</td>\n",
"      <td>11.750000</td>\n",
"      <td>12.820000</td>\n",
"      <td>8.580000</td>\n",
"      <td>0.510000</td>\n",
"      <td>5.670000</td>\n",
"      <td>18.860000</td>\n",
"      <td>0.120000</td>\n",
"      <td>0.600000</td>\n",
"      <td>0.140000</td>\n",
"      <td>81.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>50%</th>\n",
"      <td>48.570000</td>\n",
"      <td>95.680000</td>\n",
"      <td>9.890000</td>\n",
"      <td>21.690000</td>\n",
"      <td>23.520000</td>\n",
"      <td>15.850000</td>\n",
"      <td>0.890000</td>\n",
"      <td>9.160000</td>\n",
"      <td>30.840000</td>\n",
"      <td>1.070000</td>\n",
"      <td>2.970000</td>\n",
"      <td>0.980000</td>\n",
"      <td>118.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>75%</th>\n",
"      <td>80.590000</td>\n",
"      <td>149.745000</td>\n",
"      <td>19.950000</td>\n",
"      <td>37.620000</td>\n",
"      <td>40.127500</td>\n",
"      <td>30.020000</td>\n",
"      <td>1.450000</td>\n",
"      <td>15.220000</td>\n",
"      <td>45.570000</td>\n",
"      <td>3.080000</td>\n",
"      <td>9.150000</td>\n",
"      <td>3.350000</td>\n",
"      <td>208.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"      <th>max</th>\n",
"      <td>949.990000</td>\n",
"      <td>1000.000000</td>\n",
"      <td>390.680000</td>\n",
"      <td>362.210000</td>\n",
"      <td>467.630000</td>\n",
"      <td>352.890000</td>\n",
"      <td>175.810000</td>\n",
"      <td>193.860000</td>\n",
"      <td>257.730000</td>\n",
"      <td>455.030000</td>\n",
"      <td>454.850000</td>\n",
"      <td>170.370000</td>\n",
"      <td>2049.000000</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"              PM2.5          PM10            NO           NO2           NOx  \\\n",
"count  24933.000000  18391.000000  25949.000000  25946.000000  25346.000000   \n",
"mean      67.450578    118.127103     17.574730     28.560659     32.309123   \n",
"std       64.661449     90.605110     22.785846     24.474746     31.646011   \n",
"min        0.040000      0.010000      0.020000      0.010000      0.000000   \n",
"25%       28.820000     56.255000      5.630000     11.750000     12.820000   \n",
"50%       48.570000     95.680000      9.890000     21.690000     23.520000   \n",
"75%       80.590000    149.745000     19.950000     37.620000     40.127500   \n",
"max      949.990000   1000.000000    390.680000    362.210000    467.630000   \n",
"\n",
"                NH3            CO           SO2            O3       Benzene  \\\n",
"count  19203.000000  27472.000000  25677.000000  25509.000000  23908.000000   \n",
"mean      23.483476      2.248598     14.531977     34.491430      3.280840   \n",
"std       25.684275      6.962884     18.133775     21.694928     15.811136   \n",
"min        0.010000      0.000000      0.010000      0.010000      0.000000   \n",
"25%        8.580000      0.510000      5.670000     18.860000      0.120000   \n",
"50%       15.850000      0.890000      9.160000     30.840000      1.070000   \n",
"75%       30.020000      1.450000     15.220000     45.570000      3.080000   \n",
"max      352.890000    175.810000    193.860000    257.730000    455.030000   \n",
"\n",
"             Toluene        Xylene           AQI  \n",
"count  21490.000000  11422.000000  24850.000000  \n",
"mean       8.700972      3.070128    166.463581  \n",
"std       19.969164      6.323247    140.696585  \n",
"min        0.000000      0.000000     13.000000  \n",
"25%        0.600000      0.140000     81.000000  \n",
"50%        2.970000      0.980000    118.000000  \n",
"75%        9.150000      3.350000    208.000000  \n",
"max      454.850000    170.370000   2049.000000  "
]
},
"execution_count": 5,
"metadata": {},
"output_type": "execute_result"
```

```
}
],
"source": [
"df.describe()"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"<class 'pandas.core.frame.DataFrame'>\n",
"RangeIndex: 29531 entries, 0 to 29530\n",
"Data columns (total 16 columns):\n",
" #   Column      Non-Null Count  Dtype  \n",
"---  ------      --------------  -----  \n",
" 0   City        29531 non-null  object \n",
" 1   Date        29531 non-null  object \n",
" 2   PM2.5       24933 non-null  float64\n",
" 3   PM10        18391 non-null  float64\n",
" 4   NO          25949 non-null  float64\n",
" 5   NO2         25946 non-null  float64\n",
" 6   NOx         25346 non-null  float64\n",
" 7   NH3         19203 non-null  float64\n",
" 8   CO          27472 non-null  float64\n",
" 9   SO2         25677 non-null  float64\n",
" 10  O3          25509 non-null  float64\n",
" 11  Benzene     23908 non-null  float64\n",
" 12  Toluene     21490 non-null  float64\n",
" 13  Xylene      11422 non-null  float64\n",
" 14  AQI         24850 non-null  float64\n",
" 15  AQI_Bucket  24850 non-null  object \n",
"dtypes: float64(13), object(3)\n",
"memory usage: 3.6+ MB\n"
]
}
],
"source": [
"df.info()\n"
]
},
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"City              0\n",
```

```
"Date              0\n",
"PM2.5          4598\n",
"PM10          11140\n",
"NO              3582\n",
"NO2             3585\n",
"NOx             4185\n",
"NH3            10328\n",
"CO              2059\n",
"SO2             3854\n",
"O3              4022\n",
"Benzene         5623\n",
"Toluene         8041\n",
"Xylene         18109\n",
"AQI             4681\n",
"AQI_Bucket      4681\n",
"dtype: int64"
]
},
"execution_count": 7,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.isnull().sum()"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [],
"source": [
"mis_val = df.isnull().sum()\n",
"        \n",
"# Percentage of missing values\n",
"mis_val_percent = 100 * df.isnull().sum() / len(df)\n",
"\n",
"# Make a table with the results\n",
"mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)\n",
"\n",
"# Rename the columns\n",
"mis_val_table_ren_columns = mis_val_table.rename(\n",
"columns = {0 : 'Missing Values', 1 : '% of Total Values'})\n",
"\n",
"# Sort the table by percentage of missing descending\n",
"mis_val_table_ren_columns = mis_val_table_ren_columns[\n",
"    mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(\n",
"'% of Total Values', ascending=False).round(1)"
]
},
{
"cell_type": "code",
"execution_count": 9,
```

```
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>Missing Values</th>\n",
"      <th>% of Total Values</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>Xylene</th>\n",
"      <td>18109</td>\n",
"      <td>61.3</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>PM10</th>\n",
"      <td>11140</td>\n",
"      <td>37.7</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>NH3</th>\n",
"      <td>10328</td>\n",
"      <td>35.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>Toluene</th>\n",
"      <td>8041</td>\n",
"      <td>27.2</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>Benzene</th>\n",
"      <td>5623</td>\n",
"      <td>19.0</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"      <th>AQI</th>\n",
"      <td>4681</td>\n",
"      <td>15.9</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>AQI_Bucket</th>\n",
"      <td>4681</td>\n",
"      <td>15.9</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>PM2.5</th>\n",
"      <td>4598</td>\n",
"      <td>15.6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>NOx</th>\n",
"      <td>4185</td>\n",
"      <td>14.2</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>O3</th>\n",
"      <td>4022</td>\n",
"      <td>13.6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>SO2</th>\n",
"      <td>3854</td>\n",
"      <td>13.1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>NO2</th>\n",
"      <td>3585</td>\n",
"      <td>12.1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>NO</th>\n",
"      <td>3582</td>\n",
"      <td>12.1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>CO</th>\n",
"      <td>2059</td>\n",
"      <td>7.0</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"            Missing Values  % of Total Values\n",
"Xylene               18109               61.3\n",
"PM10                 11140               37.7\n",
"NH3                  10328               35.0\n",
"Toluene               8041               27.2\n",
```

```
"Benzene               5623              19.0\n",
"AQI                   4681              15.9\n",
"AQI_Bucket            4681              15.9\n",
"PM2.5                 4598              15.6\n",
"NOx                   4185              14.2\n",
"O3                    4022              13.6\n",
"SO2                   3854              13.1\n",
"NO2                   3585              12.1\n",
"NO                    3582              12.1\n",
"CO                    2059               7.0"
]
},
"execution_count": 9,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"mis_val_table_ren_columns"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"Lot of data missing in certain columns, so we need to impute. Handling missing values in subsequent cells using Linear Interpolation"
]
},
{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
"outputs": [],
"source": [
"df.interpolate(limit_direction='both', inplace=True)"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
```

```
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>Date</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-01</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-02</td>\n",
"      <td>73.24</td>\n",
```

```
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-03</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-04</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
```

```
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-05</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>95</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-06</td>\n",
"      <td>120.96</td>\n",
"      <td>141.54</td>\n",
"      <td>4.23</td>\n",
"      <td>20.86</td>\n",
"      <td>25.02</td>\n",
"      <td>26.64</td>\n",
"      <td>4.23</td>\n",
"      <td>35.54</td>\n",
"      <td>35.66</td>\n",
"      <td>6.05</td>\n",
"      <td>15.85</td>\n",
"      <td>1.68</td>\n",
"      <td>303.0</td>\n",
"      <td>Very Poor</td>\n",
```

```
"    </tr>\n",
"    <tr>\n",
"      <th>96</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-07</td>\n",
"      <td>138.63</td>\n",
"      <td>141.54</td>\n",
"      <td>5.65</td>\n",
"      <td>25.47</td>\n",
"      <td>30.67</td>\n",
"      <td>26.64</td>\n",
"      <td>5.65</td>\n",
"      <td>87.43</td>\n",
"      <td>29.77</td>\n",
"      <td>11.25</td>\n",
"      <td>24.67</td>\n",
"      <td>3.12</td>\n",
"      <td>314.0</td>\n",
"      <td>Very Poor</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>97</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-08</td>\n",
"      <td>144.36</td>\n",
"      <td>141.54</td>\n",
"      <td>15.82</td>\n",
"      <td>30.69</td>\n",
"      <td>46.10</td>\n",
"      <td>26.64</td>\n",
"      <td>15.82</td>\n",
"      <td>66.39</td>\n",
"      <td>27.34</td>\n",
"      <td>17.22</td>\n",
"      <td>49.56</td>\n",
"      <td>7.71</td>\n",
"      <td>378.0</td>\n",
"      <td>Very Poor</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>98</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-09</td>\n",
"      <td>85.03</td>\n",
"      <td>141.54</td>\n",
"      <td>3.30</td>\n",
"      <td>26.70</td>\n",
"      <td>27.41</td>\n",
"      <td>26.64</td>\n",
"      <td>3.30</td>\n",
"      <td>59.16</td>\n",
"      <td>35.89</td>\n",
"      <td>7.58</td>\n",
"      <td>29.19</td>\n",
```

```
"      <td>4.08</td>\n",
"      <td>415.0</td>\n",
"      <td>Severe</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>99</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-04-10</td>\n",
"      <td>134.60</td>\n",
"      <td>141.54</td>\n",
"      <td>5.90</td>\n",
"      <td>26.27</td>\n",
"      <td>28.70</td>\n",
"      <td>26.64</td>\n",
"      <td>5.90</td>\n",
"      <td>34.53</td>\n",
"      <td>7.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>270.5</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>100 rows × 16 columns</p>\n",
"</div>"
],
"text/plain": [
"         City        Date   PM2.5    PM10     NO    NO2    NOx    NH3     CO  \\\n",
"0   Ahmedabad  2015-01-01   73.24  141.54   0.92  18.22  17.15  26.64   0.92   \n",
"1   Ahmedabad  2015-01-02   73.24  141.54   0.97  15.69  16.46  26.64   0.97   \n",
"2   Ahmedabad  2015-01-03   73.24  141.54  17.40  19.30  29.70  26.64  17.40   \n",
"3   Ahmedabad  2015-01-04   73.24  141.54   1.70  18.48  17.97  26.64   1.70   \n",
"4   Ahmedabad  2015-01-05   73.24  141.54  22.10  21.42  37.76  26.64  22.10   \n",
"..        ...         ...     ...     ...    ...    ...    ...    ...    ...   \n",
"95  Ahmedabad  2015-04-06  120.96  141.54   4.23  20.86  25.02  26.64   4.23   \n",
"96  Ahmedabad  2015-04-07  138.63  141.54   5.65  25.47  30.67  26.64   5.65   \n",
"97  Ahmedabad  2015-04-08  144.36  141.54  15.82  30.69  46.10  26.64  15.82   \n",
"98  Ahmedabad  2015-04-09   85.03  141.54   3.30  26.70  27.41  26.64   3.30   \n",
"99  Ahmedabad  2015-04-10  134.60  141.54   5.90  26.27  28.70  26.64   5.90   \n",
"\n",
"      SO2      O3  Benzene  Toluene  Xylene    AQI AQI_Bucket  \n",
"0   27.64  133.36     0.00     0.02    0.00  209.0        NaN  \n",
"1   24.55   34.06     3.68     5.50    3.77  209.0        NaN  \n",
"2   29.07   30.70     6.80    16.40    2.25  209.0        NaN  \n",
"3   18.59   36.08     4.43    10.14    1.00  209.0        NaN  \n",
"4   39.33   39.31     7.01    18.89    2.78  209.0        NaN  \n",
"..    ...     ...      ...      ...     ...    ...        ...  \n",
"95  35.54   35.66     6.05    15.85    1.68  303.0  Very Poor  \n",
"96  87.43   29.77    11.25    24.67    3.12  314.0  Very Poor  \n",
"97  66.39   27.34    17.22    49.56    7.71  378.0  Very Poor  \n",
"98  59.16   35.89     7.58    29.19    4.08  415.0     Severe  \n",
"99  34.53    7.00     0.00     0.00    0.00  270.5        NaN  \n",
```

```
"\n",
"[100 rows x 16 columns]"
]
},
"execution_count": 11,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head(100)"
]
},
{
"cell_type": "code",
"execution_count": 12,
"metadata": {},
"outputs": [],
"source": [
"df['Vehicular Pollution'] = df['PM2.5'] + df['PM10'] + df['NO'] + df['NO2'] + df['NOx']
+ df['NH3'] + df['CO']\n",
"df['Industrial                                    Pollution']                                    =
df['SO2']+df['O3']+df['Benzene']+df['Toluene']+df['Xylene']"
]
},
{
"cell_type": "code",
"execution_count": 13,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>Date</th>\n",
"      <th>PM2.5</th>\n",
```

```
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-01</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-02</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
```

```
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-03</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-04</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-05</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
```

```
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>NaN</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"        City        Date  PM2.5    PM10     NO    NO2    NOx    NH3     CO  \\\n",
"0  Ahmedabad  2015-01-01  73.24  141.54   0.92  18.22  17.15  26.64   0.92   \n",
"1  Ahmedabad  2015-01-02  73.24  141.54   0.97  15.69  16.46  26.64   0.97   \n",
"2  Ahmedabad  2015-01-03  73.24  141.54  17.40  19.30  29.70  26.64  17.40   \n",
"3  Ahmedabad  2015-01-04  73.24  141.54   1.70  18.48  17.97  26.64   1.70   \n",
"4  Ahmedabad  2015-01-05  73.24  141.54  22.10  21.42  37.76  26.64  22.10   \n",
"\n",
"     SO2      O3  Benzene  Toluene  Xylene    AQI AQI_Bucket  \\\n",
"0  27.64  133.36     0.00     0.02    0.00  209.0        NaN   \n",
"1  24.55   34.06     3.68     5.50    3.77  209.0        NaN   \n",
"2  29.07   30.70     6.80    16.40    2.25  209.0        NaN   \n",
"3  18.59   36.08     4.43    10.14    1.00  209.0        NaN   \n",
"4  39.33   39.31     7.01    18.89    2.78  209.0        NaN   \n",
"\n",
"   Vehicular Pollution  Industrial Pollution  \n",
"0               278.63                161.02  \n",
"1               275.51                 71.56  \n",
"2               325.22                 85.22  \n",
"3               281.27                 70.24  \n",
"4               344.80                107.32  "
]
},
"execution_count": 13,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head()"
]
},
{
"cell_type": "code",
"execution_count": 14,
```

```
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:10:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Poor'\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:12:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Very Poor'\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:14:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Severe'\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:8:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Moderate'\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:6:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Satisfactory'\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\666924950.py:4:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  df['AQI_Bucket'][ind] = 'Good'\n"
```

```
]
}
],
"source": [
"for ind in df.index:\n",
"\n",
"    if (df['AQI'][ind] >= 0 and df['AQI'][ind] <= 50):\n",
"        df['AQI_Bucket'][ind] = 'Good'\n",
"    elif (df['AQI'][ind] >= 51 and df['AQI'][ind] <= 100):\n",
"        df['AQI_Bucket'][ind] = 'Satisfactory'\n",
"    elif (df['AQI'][ind] >= 101 and df['AQI'][ind] <= 200):\n",
"        df['AQI_Bucket'][ind] = 'Moderate'\n",
"    elif (df['AQI'][ind] >= 201 and df['AQI'][ind] <= 300):\n",
"        df['AQI_Bucket'][ind] = 'Poor'\n",
"    elif (df['AQI'][ind] >= 301 and df['AQI'][ind] <= 400):\n",
"        df['AQI_Bucket'][ind] = 'Very Poor'\n",
"    else:\n",
"        df['AQI_Bucket'][ind] = 'Severe'\n",
"    "
]
},
{
"cell_type": "code",
"execution_count": 15,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>Date</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"
```

```
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-01</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-02</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>275.51</td>\n",
```

```
"      <td>71.56</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-03</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-04</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-05</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
```

```
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"        City        Date  PM2.5    PM10     NO    NO2    NOx    NH3     CO  \\\n",
"0  Ahmedabad  2015-01-01  73.24  141.54   0.92  18.22  17.15  26.64   0.92   \n",
"1  Ahmedabad  2015-01-02  73.24  141.54   0.97  15.69  16.46  26.64   0.97   \n",
"2  Ahmedabad  2015-01-03  73.24  141.54  17.40  19.30  29.70  26.64  17.40   \n",
"3  Ahmedabad  2015-01-04  73.24  141.54   1.70  18.48  17.97  26.64   1.70   \n",
"4  Ahmedabad  2015-01-05  73.24  141.54  22.10  21.42  37.76  26.64  22.10   \n",
"\n",
"     SO2      O3  Benzene  Toluene  Xylene    AQI AQI_Bucket  \\\n",
"0  27.64  133.36     0.00     0.02    0.00  209.0       Poor   \n",
"1  24.55   34.06     3.68     5.50    3.77  209.0       Poor   \n",
"2  29.07   30.70     6.80    16.40    2.25  209.0       Poor   \n",
"3  18.59   36.08     4.43    10.14    1.00  209.0       Poor   \n",
"4  39.33   39.31     7.01    18.89    2.78  209.0       Poor   \n",
"\n",
"   Vehicular Pollution  Industrial Pollution  \n",
"0               278.63                161.02  \n",
"1               275.51                 71.56  \n",
"2               325.22                 85.22  \n",
"3               281.27                 70.24  \n",
"4               344.80                107.32  "
]
},
"execution_count": 15,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head()"
]
},
{
"cell_type": "code",
"execution_count": 16,
"metadata": {},
"outputs": [
{
"data": {
```

```
"text/plain": [
"City                     0\n",
"Date                     0\n",
"PM2.5                    0\n",
"PM10                     0\n",
"NO                       0\n",
"NO2                      0\n",
"NOx                      0\n",
"NH3                      0\n",
"CO                       0\n",
"SO2                      0\n",
"O3                       0\n",
"Benzene                  0\n",
"Toluene                  0\n",
"Xylene                   0\n",
"AQI                      0\n",
"AQI_Bucket               0\n",
"Vehicular Pollution      0\n",
"Industrial Pollution     0\n",
"dtype: int64"
]
},
"execution_count": 16,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.isnull().sum()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"Feature Engineering:\n",
"\n",
"We have already created Vehicular and Industrial pollution features. We will further create Organic, Inorganic, and Particulate Matter as features by combining already existing columns\n"
]
},
{
"cell_type": "code",
"execution_count": 17,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
```

```
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>Date</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>...</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>Particulate Matter</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-01</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
```

```
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"      <td>0.02</td>\n",
"      <td>224.85</td>\n",
"      <td>214.78</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-02</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>...</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"      <td>12.95</td>\n",
"      <td>119.34</td>\n",
"      <td>214.78</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-03</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>...</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"      <td>25.45</td>\n",
```

```
"      <td>170.21</td>\n",
"      <td>214.78</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-04</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>...</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"      <td>15.57</td>\n",
"      <td>121.16</td>\n",
"      <td>214.78</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2015-01-05</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>...</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"      <td>28.68</td>\n",
"      <td>208.66</td>\n",
"      <td>214.78</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>5 rows × 21 columns</p>\n",
```

```
"</div>"
],
"text/plain": [
"        City        Date  PM2.5    PM10     NO    NO2    NOx    NH3     CO  \\\n",
"0  Ahmedabad  2015-01-01  73.24  141.54   0.92  18.22  17.15  26.64   0.92   \n",
"1  Ahmedabad  2015-01-02  73.24  141.54   0.97  15.69  16.46  26.64   0.97   \n",
"2  Ahmedabad  2015-01-03  73.24  141.54  17.40  19.30  29.70  26.64  17.40   \n",
"3  Ahmedabad  2015-01-04  73.24  141.54   1.70  18.48  17.97  26.64   1.70   \n",
"4  Ahmedabad  2015-01-05  73.24  141.54  22.10  21.42  37.76  26.64  22.10   \n",
"\n",
"     SO2  ...  Benzene  Toluene  Xylene    AQI  AQI_Bucket  \\\n",
"0  27.64  ...     0.00     0.02    0.00  209.0        Poor   \n",
"1  24.55  ...     3.68     5.50    3.77  209.0        Poor   \n",
"2  29.07  ...     6.80    16.40    2.25  209.0        Poor   \n",
"3  18.59  ...     4.43    10.14    1.00  209.0        Poor   \n",
"4  39.33  ...     7.01    18.89    2.78  209.0        Poor   \n",
"\n",
"  Vehicular Pollution  Industrial Pollution  Organic Pollutants  \\\n",
"0              278.63                161.02                0.02   \n",
"1              275.51                 71.56               12.95   \n",
"2              325.22                 85.22               25.45   \n",
"3              281.27                 70.24               15.57   \n",
"4              344.80                107.32               28.68   \n",
"\n",
"   Inorganic Pollutants  Particulate Matter  \n",
"0                224.85              214.78  \n",
"1                119.34              214.78  \n",
"2                170.21              214.78  \n",
"3                121.16              214.78  \n",
"4                208.66              214.78  \n",
"\n",
"[5 rows x 21 columns]"
]
},
"execution_count": 17,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df['Organic Pollutants'] = df['Benzene'] + df['Toluene'] + df['Xylene']\n",
"df['Inorganic Pollutants'] = df['CO'] + df['NH3'] + df['NO'] + df['NO2'] + df['NOx'] + df['SO2']+df['O3']\n",
"df['Particulate Matter'] = df['PM2.5'] + df['PM10']\n",
"df.head()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"Some visualization to get an idea of the most commonly observed AQI values for the cities in the dataset follows"
```

```
]
},
{
"cell_type": "code",
"execution_count": 18,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>AQI</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Aizawl</td>\n",
"      <td>23.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Shillong</td>\n",
"      <td>49.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Thiruvananthapuram</td>\n",
"      <td>67.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Mumbai</td>\n",
"      <td>74.832379</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"      <th>4</th>\n",
"      <td>Coimbatore</td>\n",
"      <td>75.136364</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5</th>\n",
"      <td>Amaravati</td>\n",
"      <td>78.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
"      <td>Chandigarh</td>\n",
"      <td>82.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7</th>\n",
"      <td>Bengaluru</td>\n",
"      <td>86.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>8</th>\n",
"      <td>Kolkata</td>\n",
"      <td>94.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>9</th>\n",
"      <td>Ernakulam</td>\n",
"      <td>95.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>10</th>\n",
"      <td>Guwahati</td>\n",
"      <td>97.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>11</th>\n",
"      <td>Kochi</td>\n",
"      <td>98.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12</th>\n",
"      <td>Amritsar</td>\n",
"      <td>101.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>13</th>\n",
"      <td>Chennai</td>\n",
"      <td>102.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>14</th>\n",
"      <td>Hyderabad</td>\n",
"      <td>103.000000</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>15</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>112.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>16</th>\n",
"      <td>Bhopal</td>\n",
"      <td>118.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>17</th>\n",
"      <td>Jaipur</td>\n",
"      <td>121.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>18</th>\n",
"      <td>Jorapokhar</td>\n",
"      <td>122.769231</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>19</th>\n",
"      <td>Talcher</td>\n",
"      <td>124.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>20</th>\n",
"      <td>Brajrajnagar</td>\n",
"      <td>124.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>21</th>\n",
"      <td>Patna</td>\n",
"      <td>196.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>22</th>\n",
"      <td>Lucknow</td>\n",
"      <td>201.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>23</th>\n",
"      <td>Gurugram</td>\n",
"      <td>205.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>24</th>\n",
"      <td>Delhi</td>\n",
"      <td>257.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>25</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>316.527897</td>\n",
```

```
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"                  City         AQI\n",
"0               Aizawl   23.000000\n",
"1             Shillong   49.000000\n",
"2   Thiruvananthapuram   67.000000\n",
"3               Mumbai   74.832379\n",
"4           Coimbatore   75.136364\n",
"5            Amaravati   78.000000\n",
"6           Chandigarh   82.000000\n",
"7            Bengaluru   86.000000\n",
"8              Kolkata   94.000000\n",
"9            Ernakulam   95.000000\n",
"10            Guwahati   97.000000\n",
"11               Kochi   98.000000\n",
"12            Amritsar  101.000000\n",
"13             Chennai  102.000000\n",
"14           Hyderabad  103.000000\n",
"15       Visakhapatnam  112.000000\n",
"16              Bhopal  118.000000\n",
"17              Jaipur  121.000000\n",
"18          Jorapokhar  122.769231\n",
"19             Talcher  124.000000\n",
"20        Brajrajnagar  124.000000\n",
"21               Patna  196.000000\n",
"22             Lucknow  201.000000\n",
"23            Gurugram  205.000000\n",
"24               Delhi  257.000000\n",
"25           Ahmedabad  316.527897"
]
},
"execution_count": 18,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"city_median_AQI                                    =                                    df[['City',
'AQI']].groupby(['City']).median().sort_values(['AQI']).reset_index()\n",
"city_median_AQI"
]
},
{
"cell_type": "code",
"execution_count": 19,
"metadata": {},
"outputs": [],
"source": [
"import matplotlib.pyplot as plt\n",
"%matplotlib inline\n",
```

"import seaborn as sns"
]
},
{
"cell_type": "code",
"execution_count": 20,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABNYAAAJZCAYAAABlQ7iIAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjYuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8o6BhiAAAACXBIWXMAAA9hAAAPYQGoP6dpAAC1EUlEQVR
4nOzdd3RU5eL18T0hCSGQhJoEJECoUqV3pTdpgkpVaSIivYigl6qCYGgk6qU3kSKCokhooYkU6SAgUpUI0gKhhhfC
8f/hmfgxJIMyFnDP4/aw1a5EzJ2EnK5k5s+cpDmOMEQAAAAAAAIAH4mV1AAAAAAAAAAMATUawBAAAAAAAAbqBYAwA
AAAAAANxAsQYAAAAAAAC4gWINAAAAAAAAcAPFGgAAAAAAAOAGijUAAAAAAAADADRRrAAAAAAAAgBso1gAAAAAAAAA
3UKwBAIDH3u7du9WuXTh4fLz89P6dKlU8mSJTVy5EidP3/eeV7VqlWpV58dXr17VkCFHt2pQP/T+YPn2
6HA6HHA5HotmNMcqbN68DofL9/sw5MqVS23btnnV+vHbt2iRzpKTevXvL4XCoQYMG9xxvMNcSYHj+bNevHFF5U1a1b5+vo
qa9asatasmbZu3Zrg3Pif87Zt2x5x5VbAAAYHMUawAA4LH23//+rVvVv15ptvavny4vnny5eefeFFTZo0SR0
6dHCeO2HCBE2YMMH58dWrVzVzV06FDLSyF3BQQEaMqUKQmOR0hCGsQAHjksb5jKYu00Ubd4v7qvl08R11hnsNYa1cNCOcrVWgY4GjrQ5
PESUOEOaTFfOo5KXcgcxVUUwzapqVUJCNXGp39RVy1haXSqSdyj+bGX4zyPZsPNVd40CV3MbOWp1xL8ymnTa0fvqGDKzAJU
UqVMrS5YsqlSpklauXJms779ly5aS5JLz0qVWrRokdq3b5/o59/dX+YVLLlgRfL7GpoNu2bVOLFi2UK1cupUmTRrly5VLL
li11/ZXX5cqVtWXLlgRfL7GpoNu2bVOLFi2UK1cupUmTRrly5VLLli11/Phxl8+Nn2qZs0ade7cWZk1
mTJnUtGlT/fnnn8n6GUj/FIm+vr6aJKu3tXdqafmr0KSSCjOm8KfHU2Mq8aUNBljE4Ksf2+Yk8Acpb+SoFaeo7+NQkzZPMdTO
HOqdWtm3bVuvXr5fD4dCXX36JmiXPIsBgvzGQn4YPWOfnCu+QNuYiqcBGFq31O8LOaJIyPo+0VAi9pgvp9ewE6VzszGvJMdp9DcXeEG9Kmi/7oGHDLrBPYHGHP/jGFKFIi3tHd7XbagQTZDiBI8PxF0UQ1CzpH/zWIdqFPkgsLmgwmkJzNtC8M0vbZAIBeNbSMAHH2OFB/NMoMYrN7cQ5+g2RSgVzKjcnBh8UDY4Hch9Ir1ULbsVbiIxZI1OYQahcyPw9FXVhlgSeAR4dgnGNABEJvEhJfr98bpBfWPmY/jpr6OJ7p6TB3hQ8abdK1IQMiYOYppK2Vzss=
yJFau3atXnrppWT9DE6dOqUVKaocePGypIli9q9ghNUpWbbj8pZqdI5wjpTXHcu7xFKwHaQeJ5vY38NvA17LRXYttxOklpHZLwgQF
SypUrdfv27WT93wAA4PHHVFAAAAPBY+vvvv3X16e0XrH5aeeGQQi79fmpWVPKdqtVq7aSHUJ9cJQU5Yi4YT1AzOvQ4mqxH6yS+NMd6iASuCn4FN+f9fPj7xyXBE99tBEnfp12uSRGRdCI1NcWFwPwDjaqrGNHXvR1hYr0BT8bqwNXHbaqzOphqG8B6E5zmGbRKi5EOH0mvKLD1aCKcChMNPWN07rgnGOPNrunSHV3CtmiB4Fjqk/jkSuTRUhj1rZhiKPV7BYTb1Bb8yLl5vNVPqLqg1Pmh+0h85v4dLuVwoemXS0YS+uQCVVLfKs0+jCw5TSs7FJeE2Qo4Tj2RbfWxEOttq3yc/xKWAg+kv5VuzF/HQAhOyDpk1Q86PEUlS0Ou9g2JE1ECxgFdbj8vyQPmNQfdrBdE+CzoYuyvVxJR5Xk4dv+01ZJO1m2Tz1xZS9QqqGQX8eeMAvVJtmw2wUcRTINByv4xAwoaeL3sy6/IvFxdKcTWeHc0vA7TKq5xrSLuqyo1l5KqzN/qTwMAY3jHbO6ch1iHaXm+PKGmRTO+YLiSkSmv+mr9b+jBu4qDdNgDROSj8ZkCtjkF3aCR5BThtqnF/NpyEPgMWnATfWwhfbRBBEZVq0t/Lmc9bA1+eGZXJ1JKmbQ/DCF8+1sZo9Zo4LsZq0pp3Yp4TZbG9B6aNaWDS+0svP0uCPCyxypGc6X7l7GQmjjdGDpwxCpLEeCOa0yxfYDRvT5o8lrHfJ+NPEC1Mi2kAXyfHhOjDq1D4ti1r0E+8qDoVFjy6l4lXHZ9PJKn4hgbyU+tXBsMB+2a00u4xptFbrh6Yt48bZi2Rbhfmht1RmiRVPjSxwNgqMgvO7M6HxQyWEtRLZQ+SeVQmWZjjiVPKo9hOXC6yVeHWeArd6lCZ2jemrGm/Oxp+k4xYnnRlJhHmv6BLlrl2wJdzTUaiWA4Jhj+o/LIZ05egy6mD+KgrNeIsyppn1WbnhKl8xy09HBGU6wBVVpWdWfPb17rDLEXUQO7aSVeC8D7UBP4TN+eH3H1s3/Tq0DbQlHmSXmhjyG3e7T7AA/Fp5B+KTTtPqtM1nb0MTnzPAUCWG/YZ+9UxbPxkRcanNqDOVcB8M4xSEPGZTVbvTgaBg4nFUrWRSzK+FGvmpBvzLv7zM8Zad5s9OMj4NC0ee5XOHl8fbCVj/uxMDffjRXDJWxxWEK6PYDXaNxAb4pOC/SJMwYzMH9mxpl65Ab8IPeY6YMd1w0PSenQZ+4XKiThEVPT5pvZGWgPaWyKGNHq5AbpjbZy4+44UWu2XdNNh0XNzNf9RP4X0z+iJJdtoqL8K2Zjeb/Al+uEpB2OLYAumJWBnTmPTxJZJiWAtRFZXu5lR3gwTpV+lWcQdUQZ+s3B7VYFJXZsRQe8T9fcRRA8crNnzbiHBvbUcnT3+wvm3cZrM8jYUcKdV/XzY3dcz+4WGdQLrpiyVmyqlj+tnGRZWDVepNvp2XOdz4dVajvfsbxGsv3GsNfXD4dcAjo0cXC0Fm5x2J5fNJG0m5HPLIWD67Kr1dpemmbVyBn4IyomVe/OzpzXjbPtuGmefLVfuizAKjd9q/x4KdiZ7dtxgvJs0Zbf2QzpSCRr3nZdzD3Z6Whxwne+Xe3pGqmfnK6Jd35Ye/n9fIO4v8wXZWkfkXKMRbdq9XTBc2Zy0mMx/YkVkXqKvUPcB4vTc5jqu4TiUwIrOI4D5qsDcnwflPyIpy8ILHq1IVLuVLMVIG+pOrfqZUlBdKkOHwaK+8QYZ72vgo+zCIh/oUrxmDKkjtFd+lpRqxJt7azu4VQbJ+VUOWCMBgAfuJUyXxGGAahlfb8rjXdI6qzHu70NAEmlhfM3CpIUGjd8Yt+9W2GmB6KYjpP+S7aWL8Y8bW7C1EXZOn0ajbL9Pqg0DGFqTR00xm5sd+0n7YCnUfL00yymqI2PmaGhH0jajYIgnkjuzq8LF1/rXMHNpVk1hrjSyzSCUPbV8qdmHKpW6LXnfh4+vnZqbzfV/Cfad5VbaCT5xL6ZaoZ8h7Tm69r7+tjkClv8qMvBpD8tvc1kC2Y6tJ4KWNbuIVjVQtv67ym+5zkBGF3FRgx5Q0/JstwzdF43Vby3tUvbCxQm0u+cqIiIIWl2mkl6iUjbXLMimvy8KTD6mxC6mL3okl7Of2vnFJAyzYm+cQ4YwLObuJVT6WVo0ImIDqkaYDt0CKBbCoJ53DR42Flx44L+jvYgz4S3OvaWUqAo9vH0eS2UMlN3szRVrXLOR1TfXcQJ6NAG8yvQfuyMp6ZH9i+PVzTI9PVFxs2NoGAq5HB3s9jx6sbfE0vS0WTU2ObhyPjxRgvfDUK4RvJ53L3Vjy+UPOkUODg6BrP2ULkUhMnKcmg0LrN86Km3dWQ1kPVcLbaM5m+eA0Ipsec09jGm2O8y4L9rmPb+s1P9KeYSeMNEHR3W/sVmnHxRiYmyOX2aHG0Mc

yij799FMFBgaqadOmCgoKct5/6dIlVatWTa1atdKTTz6pgIAAbd26VcuXL3fZWCC5Pvjgg/ue06JFC82ZM0fPPvu
sevToobJly8rHx0enTp3SmjVr1LhxYzVp0kQFCxbUSy+9pNGjR8vHx0c1a9bU3r179fHHHyswMPCe/0dgYKCeeeY
ZffTRR86fZWRkpKZMmaL06dM/8PeVlDlz5uj69evq3r27swZkTGTRnZmswi8U6ZMmTRnzhxNmTJFo0aNUqVKlTR69Gj16NFDlSt
XVteuXZUjRw6dOHFC48eP108//aQCQKNYAAMBjrmPHjipbtqxGjRqlDz/8UFFRUfLx8VH+/PnVql
rde3a9Z6fP2XKFL355ptq1KiRbty4oTZt2rhsOlC2bFnlypVLadKkkUY0aNR4oW7t27TRixAidPXvWZdMCSfLz81O
5cuU0a9YsHTt2TLGxscqRI4feeust9evX74H/+n+n+RKlSqVli5dqjjFjxmjWrFkaMWKEvL29Fa1lT17dlWpUkVFixZ1njt
lyhSFhIRo+vTp+uyzz1S8eHEtWrRILVq0uO//M3fxqX/foP7XuZMmWKgoOD9dx
zzyV6f9GiRVW+fHnNnj1bH374oXx9fdWtWzeVLl1an3zyifr06Vj4/q9u3b8vPz07Jly/Tss88+tHwAAODx4DD
x2xsBAADgge3evVtPPfWUxo8frfzfeeMPqOHjIZs6cqTZt2qhfv3768MMPrY4DAABshhFrAAAAbjhy5IiOHz+ut99
+W1mzZLXbtm2tjoRH4JVXXv39/pU2bVoMGDISAsBFGrAEALihbdu2mjVrlgoUKKBPt+vdbJPLTZGqgwd/8z24zJAWAAbjS1WEw/sVOeOKKgJpAgJFLRvUcNPK0kA
3EZWpVbCWUMoXQJPnH5cYhTVJqpmWWzvXevwgmHY/2HRU1vexrpS9ZUAT0Ja9UMl+QDbyMJJKt0Yt+HCdYnajVbnjmuhmttvJsUvRfkpWXNdOiU2ANNFe9VoiUbQqf5Q
pZKZu5Tq9xTBrSrUpwVKWrNaANljCDVMlRz7SytV2JiZRWnIySX+bZ+WZQPb5o8RTC/2WnEAdFCdc9vPJajRPlBV8QsaXdsZRbwKubQVSGQWqrSNemDJ/6XfZKmBA8Cp1QQxpuDJtNTeqD6Vd7MqLMmtqpqutNjdMKvZqhKkS9U3V5u7SeVRDZuUXVGj6vmqNmZZ8+bRL7lmqGJaQtUZYR3xAEIK0MgWSVS/EhtHqzDjKxBHRwuVx5aSDPwwvGTnmJ2Mhhsp/xvFuvH5Zk0tGqqFYcHsu5pGnn8mbWF/UXD84XSWMt2SqTJUgQhmZjcmKmhVYIhsnmAmBuhq/pz9vRnHYZ0OrmuXy1ViC9dXkpGC5UtGQ1Sqt1IIjjkiwZnS+oyD7x2u74XpLG4mG6kb0Lrw2uN9wqXhTOrCU02VqyVGrvKhZ66Oj2Pn3n0S31TXwZBlPDw5h2hI0LjyPvWWz0tJ9nUeXwvDVTkIWB/Oyt0SjCYX8TkxBwcT/Nw3nDZTbfPHe0ElMNBxhXsYPsAy6D6zs+Av46dYmKa20BnzIpd5/60SzzN/RzQSjtK/fYFIfTLtBmfVbkLkVk3nI9+mx9aazlJmXnlzTUh1iZlP0DnKXTGpGZbV5rXxg6/50iWhEt0UuPlAy0g4mypEvoA8IdTOLwT2BJg3aLa0bRJIBUQ7htkKjbeyUa/MAnFCDlGHJwhIfeVVYm1EnWI21aGsbuKOxg9ih+i4/EGR3C8VHqtNgwD/YexBNnY8tk3DIBkJwHxVAGsywqI60DNBYTdLYtEGkFKJr0Y8A4iGWwFQy8H2VhEaAH4jtPncJ6YGShZUJa4KrqpbWp58kEFsHNbQdb4lqnTeGU7UhsBoWNqW1A+8txqqltHlKVUOi9ZsxXTUaxDJlH2Fi3jCOBcCILMwNPWMQUSnUUhEBNEOZeN0kCoAXTDvxuu2e+iaoUVDuJ3rg+cbmb0UpzSyvs1m8nZUodeomM+ERaxyOcTbWqo8bXqBUDRLWn9uYtfVOmLdOW4/bsxz/BMvj9KmmAmu1Py7sDwuHKJD0/0gq9SCiVGiKUl0GH0gXlHlY0R4T/lgQp5M5ENIu4u9lbNvaEdu6dlPNl10o5hmkaIiG9g1RL1iNW4wa+tcGgbbLWE0HC4l7KUtbSYoO9Bvo/Ubm/jM9PgzVU3C6Q3rn3GqT3L2BlyrBTFxMlaAcw6exAa/DrrBHx5W4U0zA5BuZmCYi4mPqzGJ8I5mzZ2GSUjTdxC3DTNjGttN3yTxpaydIEnHXXKFn4XBPT5j9
zYMAAXbp0yXk7efLk//ZNAAAAAAAA4F/H8mLN19dXefPmVenSpTVixAg99dRTGjNmjEJDQyUpwcizM2fOuKcxRytbiMPzbHlRFCZSFHZYVUPLrUoAMQzQnHcW4/qhtkTaFiOM5PmX4vNjtWk3GH82IA4yUMV3SJdu6r7pJWLrMR3J7l3eeskUuSeRKbqBTLGFx/FFXuVD
zbTMIR1R/AWhL9t/Hc+3GsR+q9ZFK5rK5XvbNWOfy3uDkSI0c7C6ab1mpAAm4AlX36gOWAjXfWyNrZlw/7/QJDxRgXkj5Ag8GQtvUJKkEl5Lgxvd81KJLOkqnwB/Q5dw9KZsnxj++HXIqL92Bmji0nGeCA6AMpDsrJlF0X4l1/Xu9JrMzYLjwdhqglwgBcWAzgdL3NqZE7BQVmFSJ5gB3U5hExwPF1A3V3FKgVvuUQ5bKnXVpQeQj2XOMO2KK2XDDq6k/n7G59nTgoHgmMBUYTUcJF5rU8Zg28ZrhW4t8ljJPBnsumqFRBp3ZAO0AKwWBkH1I0umyVQwEI9iAAqBcAV2a9FtRA2Vr57SCIP7SJIX+zGXhyt+9WxPyPuxySIMpKhZAKYOrUXBUMn+HgVBBKX81xkgLhW3KuBgJAY4BmkQg98nbvzH2dpmbByZfqOhBGIOvQBZImoo6BnRmFkjLFb3IfsI2yJvPM0ZYbu/tw/gJrHQ9ttdXDRzTGsrqL+QRPp0CBTpg3S1vs4Wg9mwpeCR3OLpWk2bUWNi4UcCAH59BdW1BLu5rHWIGn5Mp3I95
yuOx27wHdmoOMNthSEYPGLbQpXtLqSeyhdzfyxnwKG5fNjq3fNtuTmOyd5lRDBn3ppY3m9nTlA5AZd2LsWJmQUDkBySf1bw1evwL5qX0OjjHazYa1hP9Rg5qMzWq/Dts5wDxXAkZfAAAAAAAA4AdyU8NMPf8qsMbzUaiE45Gws0CA5BdOp5eAkRIBxxwBpzRbAwFmqJQSSGaq41HPpUngRlGWzB9qAcqeUYnxeRZfp52YCCFwJuHrVpSVBkl1CUYlYIxFiU5V12KIX7cm7iUtHZ0nC77jwJgHnpfqE/1
+7VnpOezHphhkbA4OpaxZ3PLA4vg6cu1CcK9AePP5iPCuUfwpOgjN3MKAe3Q6NAWjJTb6gTwe7lMFSwAFgXTNq3ENO4PWXbQpK3ZAA8eQEJVNAtZsQ2nF3bNjhQfEHhCvcDRjjbE9sXB0dO1/DBdQ2qQ8GzKLexzH3Mk/jUg0CdfV7vD71GZD3e8Jhg4FBsdDKIpHUW6p6pE5SAWcHrb5fFXdpuwsWcsQXH0bRPKfQcdtgyd48sduxZ6tOVD5TTmQO0gqYVu3JCH7SAzqlAlq5V2JBm42nkG7oKeu3J1JdaFhO2iGkMafFBa1Wj9Z5KhPTV4H40nY4NnGT9h8Mn9txnUdj2+xGnzvRXbHvA0fdOd48YWmTUfHdafF7SlpqfNJ3+v1gZfdCqJfYqLqYaHx6bmyUD0Nwz0FmIOzvmoSYIRv9jyJJN1qXh/BfmPcPsT8q+8aw00YYFJJUXMfB7g+1HOf7VWP6tA6ggRKwCebMUL7NA+ppSkyfwp1SHQb2DG3AMcBs+DnOoP40rvtuQYZmnrXVQ8NGo8JNJ+Zxv09V8JKhHsmHWq1DUexR3AsVTIAibcRS1FtmpD5skIS1ObBsPRBNd0CZxngY5YA66M/JMKUo1KD9xU2HKk4Uobhm2/wMQNq2GHlAJxBgXTHu/9s3ysCBS1QwtvHSCFxHf4hNdwPPa1bo0ByeN2EU08WpjyN00LVQpRdPUzMZWyBy5oyQ3jBI8KzsYt3uwzqEnB+JJH5sZMnfsAh8fe8WLVxzdswyMz2w4D1MymOeWV5BDL/ZdWnqKmp3WC7cp+OwfzZTsoAOhkUywQqsgeyuVm1eOtwkmcaLHNWDALtLgLy0zvFpUYuzrpYmTAt06u76E5QAAAAAAAAAAAAAAAAA
yh+GOx74+fbV8Q0kiMqJJvuMD3Iu6rOE3rTdPbudjIqsq8YMqkqhxRD/pUG6X+gIQXqrjPUGa2H5jQ+MmXlq5zFNaEmkVTa2C2JLuaYVYLcjlEqkzJhxTKmqOFEZB0UkhQHKRZPLmYKvCxI4XPv4vKwbBYcHnjUCcb1aiyfF0hmbPDAbnL/iROTmUmF1ED/Xe0tvYR0EvJJp6Sm4JwOYJnYWg0wVUqu2txpZ7ek/EVr5lqSmCUlA7mr8lYRyIHb0iGqmuMJTYCeJ0Q6kSljjWPa6V4QInkiSx3c5o7YMKYrVUYfbM8vcvtr71UpnjP0R9GNvnv3G9T4wvtLVGOXaiDOjURCKJNMLOv5vVFl6UL8kLGLbZeQ1cuZZBx1oP2o7cqdCXL6hZdBtD4wJAK7LUw9f1izC2JXmwPT4TM0dIO9kC4HUtTQ2TIaZhuFtt8UexRdt4PXtbWtkGsbd3xmrFk8VXTpd0cFbmMJCZPYHmMqg+xnkE0ZKGJyiBkDWtDEF0UYgAjxQvMpQ+mq/xyDVBdE3E81FDjyVEDTnDDBRFCAZF/5Z9UHk0NtL0C9GldOOMm+WHjN5nIwXvEx85Ag/bt0L2snuLjYfmOTWhFS0q7fpAh+SG+GC/mVcuzFoEVnv5dktRvpYCCKhoSzGL2fHDI+CPKa48EqrvDI9FZD/l/mI6tg0Dfvx8HbBhNsPr58zIhoOhUr5v8zmzjkUf+OWfGlNtaZtK5XhoSijK2/3+EUSBzq7m+iMdQuWfbBu3I5/pFHZVVmW7L6nZGPyhp1f9UNGSsglXQ1uwoUCqj/Zk1dNW7rJrdsp+u1HJsvSRNTOnnUsXFxbURlTlgqpnkuDaYyv12ApmDiAyzb1F5B28w6xKEJvM5CJEVT2W9ppcjuFTWr71OaXJajgP6qr1Vk8SnoJrI95hP5JDP+F7Eod/K7fN3HsY0p9yQjMvShZmNfTr0Km4ZBUzP3YnKnRyUkRMd2HAl4+JnAnf/ybfSpw0WyXz43I9/LXSWKt08lK+0ZS9uXTZA1hQvx0mZVqcKCQ8T1hyJzlWFsBlpMCU/MMh4hkXsj0Q8Y+Pjx/PY1+Y8+RdvMjtLdm3Fi6uq2pfOcLk4Wxc5W7s7ewCJpK+FyjUgUDsMpfPo/NLqQ2E9lGaStAKzM7bpO4wdkLj6ZFakTLNxSIZK2Ki30t+dNSkw/EJN2Zr2GqalnTe+I6Ij/gqGZ1qiAjZYjzJG3Fje9tDRpiS71tgsk23o1q3Wm5Vm6pK6GPsNtWvHmZe95G/9YPe1u3ApZOhVp61w1nvXJmwVwTi8sPJs4Ev/XHzdnpwWkh6q8Hz2qeNNr9GnUd4nh17EaJTpyWqmbKhf6Ij5W87ZzDDT1rEzd3eyiezVR9ovdVvgH7+YMKY3wGHWbQoRWj+vk2exsFOXDyIm9ueGNunPqgQ4NFamyf7sDVeb7mhBh1YzfadqovQX3HSVO1NhUIwzRPx/u95OpTm4GBq0lqi1d4O+tUzpWSUmSF2E3Ew7JxrxqI+cWv4yF+h2jTwLsHbUhvFbfzk4ma4M41NJC6SEiXX0LCdMkCDdkTJyO9NtnClpmUdSDH5yXiZWNzJIWszP8zNRumdFU1TEtjuNq7B5THWrDSPGbOcu0iqVqPSKkLF34mJfJQVsYQKa6qoj7RbVf1i4+Mce+9hE8YjxAUpmHRbOwbo8xHO5uOp+eLd3hN17UwzTbMDkhoSc31CKXe5jNk6sK1E4E25jN1Yc6n8lHnXtWmKtQ9JfJx0I3OHjZnu9b4xevwLaGHgzn3bx4ZsqU57Ri5BhArvHt5tm2cnD5g5TLGAYI1PErNuzbOO1jSeKvuFFBGwTPOj/ZbcWlNFfNzKxTwXBVZY5rtU6zbBqOVTadNbp7mvmBYfw+ZnQTMNp0eaVfBtZwDPaanlh5E7a1q2fLQG4eTVuDWpNAlbsO0kA6fCwQgHv5QxNx5+MrqQahwzH4u/u5Pzjj6GhiUDHzYRcrMqqcRSeCWyCjRFY13XGZgo2wOSjQOGtOwZb2q0/jlw8yi1PfU/K8E5ZMKXhAbp6ObWY8VT37FuAsRdmcwGOzmgmb3N9oEyPt2mOw6nPn2iTiLc25EyygtSFnvt0nbnUqTuLkYCOJYVB8+RaSeIS6q3bUqPxXxTXH+T9f9XdkN3X4ZmkKo3tJWg4qBsUHz4KuU0k2awFpk5NRX2mSc1J4a5S4zoQQJX3z4ZLm7wQB3+aAaS7ya15Y+xU6TFdTDmLiUy+bUzMIEdhmuzwT8zJSs1gCiPeFqBbRs5bAXwF0AYJnR+3aSOUiSuRamMixiRhFXJVhGpLc4aKlSgnV3Izn/qeoA8UjNK/iIZ+Kz7pqt12P3D9XkRM5Yrx0qUvd19pOHfvzFs25/T9hAY0SMmUJXrjg6bUjn6wDjjVl4xXW3y1cIVmq6MIeDB1MiUj7aVsVCrsjjjCwS+e4gyNY/+5k2g9JZUeLR1S/RJwYeA96DjGNlhKSLsrjkpuAAIhv9Zp6+2IgyOzD6rX7E9LdnRk3hpfijI4AVU9lx8Vf06y9dk8lKAo3iRD+5YXnvgwLt+jq/5Pb3NyPKHH5LgJ8mrvZqT6EbX/vEPZKfl3mbYfeDU3Pqa5vlq0KHu6XozFLRNnjlqwnl3qGt7MgO3kGeoLL1tAEaE7WWE8yr2o+AwMGCGoFSoHr6i2U+O0NvgWfnIQvo+1HbFnjpeLH9qZoZCl7dyMV9fxS/8YnyB2MF6UBjm9IKDqKXXbLEpORY53fDk6Ix2n2Z7KPLWK0ZP9VV+qDzq8sCGP0sAVCwhEsGpvSwM+BE32ALbE1gAUT/lHv2FBbDo3WsYASzTfoG9cBZWA7Vw5yqiE3Ub8Z3tsuNEjI0Gq1vQmRZ/B+aYSzhCqWWI2p8WpoZbZ2fnb5fYByjEhddZpgaB1fCjqASQS/2zljdUsGjBJoBOk/CjXddemplRQaBOV+HA2PEyQ1m/SL6p8hWtx4QUcO31vOwI5SY8kl3zp9Pe4QKx+6TpjY5Sh5DxBsvDAOZU0gM3N3JllNcl4QlrAJaLgp0WU2aWWXdnGAAAW2QYZyA/jy5xA4ZfGbXRQlwL6CnL6Wqm76EjOw4ADpETA+1yVwrYJvW2iDZGlbLrzmXpgDXWuzB/a6NeUHgPUaQumpb7YCtRCsUcgaYHAuUpWm1QvKS5oKgRrJtqYJyg8s9KERiRhPOwG3V0WqgyXNHfH9mnuAWUNRrixr+xQLnSzfX8ke1Ubwm6iknImA9Tvab3/rv9dzLkGiJAqW7SCpCqYJP9LqVNnPFpLBsc9nRKIKwvBbTb/J/CQaN79u/Jwr6VVC8c41ZnOL6WAF3dW82M0NplYEBmt5SXSr3kQ6KEoMU4lL7yPBhBYkK1o8fZBCUYCF0/SC0VGgjmxbhS3Cj7N5eRgEuZKGY+4tJeuPhDGVmJcV8YvUIWgH3fG4JF0FoXZYTLj5VtRpe0UAkjNeMtjIUrLHL4bZ3y5RtSi4Cs7HRE2dbqsGrKkUDa8LmkZvDfzKnRMMYVwW6ofWCzKWtbs6f3FT9BbGebJmtbhzfDhuo2qrbmA5XMXmxGPubkZeWRtJFzt3Xsc/u0NC0yYfIrYG92ZfbYRwBfGzA3F9Vnn0SXnd7sUAU7PAH7lDPwOmNqOjl55RrcxIwYB0sqyQ1EJCUYDIJjS1bqTGfYa9XElbRD+rIm0MQdrnmuyxDcKJKQ/JJv7bSGtRUpKp8X6MNyPsO9p6+bTdQ4PIG5kbHcfFuezpDDycn7PThSx2zkM4zJDJUIn3DPhnrZ1C9yzZbJlBz6tZ6oSrUCpJdB3C0sErgGGe8zGA/cMsf1jSG0aSf1rBvG3cqkBm1oMPgQBEqzgt+M6yvdWPzF6fFaCDMaQDp0VtA7SLO24LRqVVNsFuiT8VVPo70UqPdMzLFmz8BrpzhXRCV7P8vA8NkjKE1W0P3EzIAmVldpSgAPrbMKbJ2Vr0M2A8Fy9/xL7LlczbDbHhJZjp0HvdiWzNHXNEF7YN37S7+vbYHScv3Pr2/Q87JF0CVWQ57Pe8SJYtXmpkWr/zXz0Tiht39Ga06Uc4iG1w6fbbtHqtLMb6JbLLyhPW3bbTGu0HDuxDh+fbJMXlU7qlrDTzO8dp6DG/4Mqsb90P8oXnXrA4qG5ltKsJK9x8MUFSdqKeovGc2PxXJOqJUBSUGRy+U7FCo4klX5+NNZVLK+Ra+HMxKnJgJYDymXKFhpo1dOCyqgNdt/pbV/Q2eg9eb9SQbl1frEZlJp1/Amwd6HddRQFp+FDzyqhKg6/tfmPG7zuZ5dRr5ThOoLbF3d5GSBr5NXSBvddBCYhGo1M6fW/IBLkkFAjEhgzc2wT0IxxRqmQFi3u3HHcB7Lz5KtvEzl8sTX9q/XVz59zlQ4Wm3ZqbnQfOV6ezR/Zwv3cxaK8nvgvzFlRdGVxhNFm2jyvF+MV9NMuzjNR2zRrdxH9rRPMqpY5sSBxnoRMu8fR4ebSMWHEQq6yq2FAXqo7UCmLd4IowmBDS+uE9AEl2WsUP7wb0f32RYQ0WKgLlNFF3k6wOI5E/a8Kds4Ni8R5TQY8TSsxHL5BV5GnqYbaLxIPPH5ZFKsHQd6IKc1oXO0d0JKhwYf7f2U0Mk3HvWIWy/pnwkqrhvZLFBeq5QQNmgKD2R2/Ct3MqpFWtaVs/MYl5IZQRSdgVeLqgkwRhfwSLyM4fJ6TPPYNFBb4KAWnngyWnEQ2cHq7Rr8pBgNTMNsZBS1Sj0AQXrZiXXzHzeHf+iJPfdiMcfb8AsLp4wUXxAIjeF1ubmE5oTCqr6iDp9Ji20xwIxPtKWYbWHrTAhbTg3RIHh3ndyZzDEOtQV2APHzK9j2F+TzBwtRm4XvCk4TuCGsQLxmkh6jHY7yOr8O6fNGb/fj1d9n8uKYdSvpC8BEHQqkrpwN7e0EXNoTQKvWkGqbf8tvJ2v0dAiiNZwUn3cRPdAqNU5zYGy2uVXuQwGJKP9KeOU2sGfzvbKBSjdhNH0c+WYAdkbtB1Zo8nOiGGXcG2FCXWlsFeePqdgrBPyZsDyBFJ+5lzSRdBnxCrjzrxUjhfXuCEUVBn1ZyWzBxHqqHAz7xB7IpuUWZbjHA16YhuxGlPzSO2+3dqmTNrzlEtKuAwsvgLwgM5BM9zOy8MBcFqtJGhyMsXOGzzzLCkxhOOEumVqKohFy1cMUJ/fQa89kMHrozfUzUFvsWZ1U78VZAgc9w8kKvu1jkuEjEIxy0Y3B3ZXBC4bUCMahzBClyGf48iL9/AOGnyWXyMbHxTzlCmrC0/nq8lG2kAWUpcjw2HiGh6qPpDkoVhYLNm4uUs+Go+1eUPD9SG2A+yWFgMz7ZeUtaJbAsw37OwXXUdMWDyMp9lFbfiYCKqvh/AmXrIcl9Ua1SsWLV2Cin0FjqJj+M4L2qv/ARChl0+TtKXcv+SDIkScn0i5AEO3a9kVSVHaEVJN3jzPn35+KEW2z9Xhs0/ji0YeT2ecm8PJr7lJSzKI3PY3zzu3CcXZKQi8tgfgXZ/RIDvR9OdLzJupWnhYx3Zf2bX0t8PmZ9YJuxvHv2vzI8eZ9GaNvRbDPQ25DrJvqHhZUgKuhjW7o/Iuo3c2gqL6t+DKgz2/0m77kLmTV6KpYHV/73YKVGTHKcC4vPsakT0G97W0aZ1Li1rAEm/z7yH+VvDmwvp+3S1jP+fz8Bso4iknDUg3Re4zwShyR4N3UwCMdoHCbH0Ui8P2wBxWexrRI3B8RUTUbm6Bf6eqBwq3Yi4bxuBt3/t1Y1t/mq5BkjXe0Be3LOJNjdNxjlkHVBhfhbFQR8lMAI6fNdOkG4H5SEkMrEtAThVMz1N5VPyw0SiAmHYPC1tqR59HDoVmb3aE/95+MAU3z0ru40TBr0m8W3+5gNeVypqzrZNcbqcg6KuV6BRAqKrsEX7xlHGvrmyiQc/Wx66y28QhxAk7WnS8YfNq4T2K2rt4a8YbnPdAZeZoD0ddSaNM63+qBqu5Pk3k3ljmc86T4NUs/sUuQovLkJzXHx8GfjcdwGtUgx4Mqr+E6ynUg9JDrwFXR1NqYd0yCp6gwBmhEgWwQ3t9TaM9Hgh5jVdS3UGcvV7PK2tDeMMBuSvyy8oCjU8u9rFk0xq4qfAR1fv6apNtC+U3L0zWYuYBTUWKvzmQSJ1wXtHkUpU5nyqFMuoVq1MfmqgBtKuMIsUnqwS9XKa7qqmL0i1XvY1KC+QtmG2WSSSl+E6pnNLypWUXNNg5hpUUuxqW7mIUVBtyIPYUFMSXK2uXgg3eVn/6rZP8i2Pr1WqmaORkbXdYWx6A0q8bPT5ZYLNn+5uDJLC/Ug9qGw80NYHLMuUo/sIGKsGczEMOf7xOdtHWh/MzqEMBd+z9b4fuQx8ZVT1ClSN1A4TK0WLK/3H4b9t94FV31MmJjvb3Hq/dnBVpyR6B6wL0P6bbQVzb9fYxIXX4HwAHJvGV9q7JBoBtUW6AbMTKnAHovIWCK7AXWaSCoG2b0+OujPMoxXw9bH0fVjCjLVdZlW9lrU6xdl1NiRwdnWSpiA63wDOTxYB/c1Ey7s3h5t7QwdGI8LfuIJ8SmhFV0A2b/lArC4s9V2tkI2MlJxeR7K6T93AiqcgB0YuxrDcC4XV4c1cpA5cF0r0VPchp+Bi6f2Rcv5mOvz2XKNHe4iihvbndVxPLW+OpAt8mFsI5eg2Bkc+O9AUr9KNNZGsO/fzrjSrJBh0nYEGw0uaRk5HFfROx5yyGjWUpnvhxKyS0ZrRF55L0qO4Y3Xe+TNbr28UxRP+aD5jnPJ9kzb0ntRlQN+W2SDwJvnTbvv6YwJEKoJr1ZDfiYOcM4rt9gVnvqh7Pe/bRRm+0rHaNa4i2RU1FXnp7DYWIDxLeV4XWyE7h6HzUwFEBl/Zh5PuyYQXlHTO6RQA7jC8nKGTHqJ4xh7kaFLh8eDcnbMz9AWV9jKC3iKQNOdMGTyFVXrUT7O29pyDcrQ9WR2CHZZDx3y8ruWMGZfVHLb2wA8+cyU2EZy71DbmcDgNC1qfYqQYfFAyLUWtLL7IMnb3G5tQ6rf3UrGXxUKmX+I82ZbuY5VMHVHh5WRGTQLa/lCnZNrxzrOnKjFeacW4t3mg6NLCkh9SlKc47R3tZNF24fd+F5ypt0a/OsVSD/oN+xaF75xmrfGBXdmDHTHsJaJTzNyz5KXHllOTUMZ9sv6cZGTJWsu4H3MVZrVBO9e9Y5KMDb4aMunYHwI4U5wz/ynEGyNmqkdrG3VeoTiAKW2i43fxTg9bfLZ0o0C/bGAeowhDYIJc39KPD+BoOWBQUKh73O8JnmuPMcyLBpnq1kRTYFavSnr2h8HTZrFDrE3gfEB8z5ohbF19Ylz5PXjUFbc3htFrPi/zHhs6y4MLKvStLtCSsJsqOe/G3ceN5YnxOIrdC+YDo2f+aXQ07BhZ3YWlNtDS+ALKNr0mOu3+zF7qZHqnV6dOqXpsvvQDVvC3lYFlWNjzsp0RStMkzTyPo1FmYlLO3mTKvebY3eLjMw7VlM2j/bhKRDMDIHXP7jxgsvjxd0wmr2GEoRX9xG4rSftDO4oCfOOBYJhp2i3vvk3iOjU/Vo6j1jzQB1twL0yrNWJkGtI3L7UVk+AyFxlAiD/N6yl+o7lJc7cC+xhEu6MUcgwUlb5ZMJsOMcXWHM7Je5bTzj8qmQ4vcgdGfxj8Tszn6iRrpsoeoiDe8xH6tvm4rwVFYPSm9EgCCxXjScf5l61c6/QTHyt6kWYzXFXwbOPmh2P3ycvl4rh6mDBxaT+0aDM4cw4Y6CjyCgtDMRnlqBsl0pGbbyD4oQXbQO6Ea5DKlVrbDNwHw9eg38ez1gKZnzyw6MsTIWG+ma4CY+JZwSrcsupk8Ih4zRkk1bqI7vrjqPmc/fFm4ITIZk7FzOxhgVJJ9Ir7o3NQIGtukLZZGV0Mrd+lsjBhZI09L8MaZ56k2kgZJc0s4cOG8eDHAmMA/NBnVQwqmjg1yYCr1eHyzTUZ4PnE6Nn16RAJZsrRaJlq9fT0z7e6d6MvONw/ye7M0g2XKsYZ9EDxKcz3cbsHzA1eWO2w5lGBxrYvBPNTNJl9JQCoPWUeZB/d7RFNgPKzK/p+yvaQ4EO2RxWrtGN/EHcxhrQIQvytlrkk4pJxaYZxDPWJ6fS6Hgc8s8gKX5nzbbM1TiLcq1NgZJ5p3rb8X96Dyg8l9I56ZVcSsfajbVjdSgr2J/KqPBFuw4KSqmUuedd1bd1H1vWyI+rwtz6eHPO7vxUCPiSS+hy87ULa/n9QTCpGhWjLUV+U7+W8ykxjtMXDM8xIybVqt4fp3KInhBrX6NBkVI+PNI6z7dV+rmEaY21AW0DxZEJbTa7zGA/4YhZZAqCPXFWn3YhIzRPnKqmVu7dgdBsgC8jsF7Z04BMHPGVXGt6jw5MJtWYhZa1ap4NqtyVIzuRj2+PFPxVGIdJ/IqwexMuSu+KG/MfnAJQyJgAWCr5a+m5/7r/pWAm7HmOvI3X66wE6SzKLTBWDo1rBS1uLO1F6uEgk2yEbGrH6E7Lh7MKhChv95uDjR7hxAqATYKmFoCw4Ozp4LwDH2v9g6oRYzGLIgFqShvpvXWFnZXr8l/3AcmZ+3FT5WNWhBmadqeg3b02xTKC7SByWwqPXLNK/lyf59WzqAK4vwz/QkA4cDEeBuLs5yTg/2Q2H/PpvvYAf1fwYKDPF9z/CrDKZUWq+vNgqi1NqG1E0gfu6HMGtg3E73zGpvsFU+9LMhtYkRaoKSuqv9OYJvM43LoL8T20qvupxiyYRhmVedF0PmMmYb8xSHXSYuXZAkS4pR2aQW1ajO89qnh7ThG76IodZ8tqJr6mhlYtMjOI2DVnvNqCOFCMuqufezX5y+Rr9vt2Q/EvdBPRKmR1SB+UHM6YNH1RHIRtqdBL8fQsHdeoZtx4Evy8xGGbjkGjpxzCvqOMGktB7RjAxWdSHaZnYtEvWBkkhIzpgzZ0ZzVLiT4K9H9ba4+Nhk6sRSAlL2Pl4Ki9v9w3I3wybM5H0nuYGmEoFo4VvHpX9Mz2PQWSgEOsR0Cc0hdWVIC6VYX6r2WYVk1pZmvgy9aYOl6xwGi5wpcFv4RXFhGb4K+3TptKb9PEsH1cD4XSiUFWWXRVx1/Uz2UYwPJ9s73sQLkfmNWgwIn8KJvHs0R4rB7e7Mz8yxUVZTJnfFkdJCwWTSpLLvL+CIzUG/7+DBC6qzZ2yqhn84K7QBtyuxdfvNvjCCfcN0W5VYcfFcE4hDW/9pB1xUMGv00cIKv2jZOwGqz1c2V5ojiX0/sLHUZD1T1N3sf7GRA3vHQeBHLtSOyh+y74ijDFUg7u79nStZsH77h2MxLuyZ7yzd1P0vq29LNPuw2UcJwdxC0aZ9GZwL2DDYQtpX9QFBqfvTxS9ZtEeXDgMN7I0+I58Fx9A6lNp9gL5kUX7OhDtWe6zSU7f3S6KeHr5+Ly2MkFd+QkiyBgk97XAi3QJE6NjFodfpOHm6m8XJGtZS0GYdMAy4NkX5EI+wWMIlb9fKwE0Z1KNz0WhUjhtMaJmUx4wlVIJRaEdB6yjxqvVHt3oLBsaJm9f3Fs5BTUaPvGjc4OUxpgKovaHC+/31nYbKG9zz7MEaZ87zgXMUy0Fwv/FAWk4+ZmWqS8DNsIJs2ZrtYT7PP3whXcC6NSPmY9F93l1OCBcIVsi6LuskQMz+J9sk4LxPKzmmxsBxh4mgc+vl2uhXK1nrDTLtZ2RHCIVXgb1ctfvYAQF5eSPt1IpDWR2W/iZo1HMZsVKvP5olfXQBRFA99IzTtgSxdrNiAC3jxk0Uw8YV+HHp0+dyRKFFnbqzoUxLvcdr6WtyVfhvhRVX1b7i/AnLh7vpmRt/QmA3/Ehcxvtmq3ihuSrhc02Izz9n3uE7+m1KvAa8DYEnJrAqb6mTZJIVuPRYw9QX/O32pzPVo29C5i/uMKl9sKfKiILCRCDyMDh9Jvj81hpeKiwIC8yfAbUBZjg9c9NaTa3xsBiB/rVQIqxPVNUH94lR0XDE5z6OZL+ye5K14UwlmmONCImIRv5BKdZgW3J2jFCNc6M14C3xoY1wdlQItUc0ezdvvIPUhiHDu2PcuijMkYLx0AQAu+9ur6TFCcHAnLmnoVxwTkzgo0L1nrMVZKwyEOUv1S9BHM8WplqFmrU3NMU8yLbpQcNfaG9HfPVM0+GF/hfWJ+eX5fqi8w3ufPzbMIzjxaxa3DxkEufGKFmsP2hEwPpFBVRz6R2TxKofX1uSXmH6YdNFh4PL84z0sQ4ACjC/TjOJdLiHTeKqwXsqqkPEPIMkZHRbf38KH6QFIJoUl9AqE7XyPN6T3/7K8uaP9SeUK40bsd4JpAFkjaJVK+Zx5ZeySCrdOxgkGwz/ttiCJCM5tVTZNtlYQpNEZxQRK4lCTqBsiwv2lEhmUJg0eL3A3sOLlYGLZH1GTiKN2p8JBbN8YWVDFMXsTJEPGlY3ZybcYvKPQMNdBAAAAAAA
yf/kqKOq5mjz2u6OBhvOO8oCLZ7qhGmxfbfgRN8kaVU+Cx8wHjftvwIcOiawqHVZ7y80Kl3pfbtEAAAAAAAA

AAAAAuIFiDQAAAAAAAHADxRoAAAAAAAADgBoo1AAAAAAAAwA0UawAAAAAAAIAbKNYAAAAAAAAAN1CsAQAAAAAAAG6
gWAMAAAAAAAADcQLEGAAAAAAAAuIFiDQAAAAAAAHADxRoAAAAAAAADgBoo1AAAAAAAAwA0UawAAAAAAAIAbKNYAAAA
AAAAAN1CsAQAAAAAAAG6gWAMAAAAAAAADcQLEGAAAAAAAAuIFiDQAAAAAAAHADxRoAAAAAAAADgBoo1AAAAAAAAwA0
UawAAAAAAAIAbKNYAAAAAAAAAN1CsAQAAAAAAAG6gWAMAAAAAAAADcQLEGAAAAAAAAuIFiDQAAAAAAAHADxRoAAAA
AAADgBoo1AAAAAAAAwA3eVgcAAAAAAACwk6hP91odwUVo7yJWR0ASGLEGAAAAAAAAuMHSYm3EiBEqU6aMAgICFBw
crOeee04HDx50OccYoyFDhihbtmxKkyaNqlatqn379rmcc+PGDXXXr1k2ZM2dW2rRp1ahRI506dSolvxUAAAAAAAD
8y1harEVGRqpLly7avHmzIiIidOvWLdWuXVsxMTHOc0aOKlP/1U48aN09atWxUaGqpatWrp8uXLznN69uypxYs
Xa968edqwYYOuXLmiBg0aKC4uzopvCwAAAAAAP8Clq6xtnz5cpePp02bpuDgYG3fvl3PPPOMjDEaPXq03nnnnHTV
t2lSSNGPGDIWEhGju3Lnq1KmTLl26pClTpmjWrFmqWbOmJGn27NkKCwvTypUrVadOnRT/vgAAAAAAAPD4s9XmBZc
uXZIkZcyYUZJ09OhRRUVFqXbt2s5zUqdOrSpVqmjTpk3q1KmTtm/frtjfrtjYWJdzsmXLpiJFimjTpk2JFms3btzQjRs
3nB9HR0c/qm8JAAAAAIB/tdMjT1sdwUXWflmtjoDHiG02LzDGqFevi6qChJUkhIiMu5ISEhzvu
ioqLk6+urDBkyJHnO3UaOaMGKGgoCDnLSws7GF/OwAAAAAAAHjM2aZY69q1q3bv3v3q0vv/wywX0Oh8PlY2NMgmN3u9c
5AwYM0KVLl5y3kydPuh8cAAAAAAAA/0q2KNa6deumpXas2aNcqePbvzeGhoqCQlGHl25swZ5yi20NBQ3bx5Uxc
uXEjynLulTp1agYGBLjcAAAAAAADgQVharBlj1LVrV3ddavXq1wsPDXe4PDw9XaGioIinMdu3rypyMhIVax
YUZJUqlQp+fj4uJxz+vRp7d27N2g7/2zOzVW8T3dNWkUGEB+/fh8fLn9/f7Vq1cp5bocPVu26Ld4a1ltLXLwwCHJXnV
f+kYuQ/yyvvJxzzvfhg8fLn9/f7Vq1cp5bocP/OHdSnTx9lypRJGTNmVN++fVW0aFH/MqsaQKk/sdCvN8MNY7k
2AXMLLe+Wg7n+mVmIa48vV/7LOvztKnTDS+c9lRtfKCaEDrcxZsjNWWP3SWyNGEqkEJC2Zw2pNPASJbhSwsk6
tUl7pl5ViUqjYIiYLTMn52V+qDmVePPi3/MlfSenHsalgr+FIBVWNXcdZGIKBBoGxLpbGqdMxAt1MalIjkS0KY3nf
ArHOoBXnWxkHM95ryS6RPNswGzX/B/vHOEUCvJNoKAtNyUeo3O+KbWZtomOwIkGFQrObXaY9Mt3OEGbu9rEMtSP
E4POBdrt7WtFqo05pS9/JVp/KsudTxWZqEwf7x4JXgOXjQiKWCmQnz9VNWG0oxjfNOTHeSAq/A7O8GFRGl8AAi
K/WBDGjw7rWW6l4p3rmU2xMuP8RLNJsCCZU00eVo0AFuTgQlSmYrlMzlPaQ9IkcHqbzUfp4C4Exx+lIzQgZUDZLa
cRCYA1m/Qv8x01EuXvc2BGaCdbqbZnlDbZ7ciP5BUTLVwK7Ehoz8qvBcq9VexhVojCPmxl4jxbS/QCsEuGFcM1Q
AAAAQsFAQAAAAA0bUUDmMAAAAAAAAYW1DMAAAAAAAAAQ4BEQ9AUAAAAAAAAAAAAaAAAAAAAGmAAAAAAACgtAgQKCAx
AAAAAAwKCzu+AAAAAAAAAzcBJsa/AAAAAAAAAAAAAAAAAADAAAAAAA4OIAAAAAAwAAAAAwAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

ADRRrAAAAAAAAgBso1gAAAAAAAAA3UKwBAAAAAAAAbqBYAwAAAAAAANxAsQYAAAAAAC4gWINAAAAAAAAcIO3lf/
5unXr9NFHH2n79u06ffq0Fi9erOeee855vzFGQ4cO1RdffKELFy6oXLlyGj9+vAoXLuw858aNG+rbt6++/PJLXbt
2TTVq1NCECROUPXt2C74jAAAAWK3hwsVWR3Dx7QtNrI4AAAAeEUuLtZiYGD311FNq166dnn/dnn/++QT3jxw5Up9++qm
mT5+u/Pnz67333lOtWrV08OBBBBQQESJJ69uypb7/9VvPmzVOmTJnUp08fNWjQQQNu3b1eqVKlS+lsCAAB47DRYOMf
qCE7fvdDa6ggAAABOlhZr9erVU7169RK9K9zxij0aNH65133lHK9zxij0aNH65133lHTpk0lSTNmzFBISIjmzp2rTp066dKlS5oyZYYpmzZq
lmjVrSpJmz56tsLAwrVy5Uq7wUAAAAAAAD/LpYWa/dy9OhRRUVFqXbt2s5jqVOnVpUqVbRp0yZ16tRJ27d
vV2xsrMs52bJlU5EiRbRp06Yki7UbN06Yki7UbN27oxo0bzo+jo6Mf3TcCAABwh/qL/mt1BBfLnu9odQQAAACPZdvNC6KioiR
JISEhLsdDQkKc90VFRcnX11cZMmRI8pzMmRI8pzEjBgxQkFBQc5bWFjYYQ04PAAAAAACAx51ti7V7HAkNhq56PJGqCM
GDNClS5ect5Mn+UrAAAAAAAPj3sO1U0NDQUEn/jErLmjWrWr8/ZM2eco9hCQ0N18+ZNXbhwwwWXU2pkzZ1SxYsU
kv3bq1KmVOnXnXqR5QcAACklPpff2p1BBfLmva2OgIAAABSkG2LtfDwcIWIGhioIkIlSpSQJN28eVPRk28e0VORkZH68MMPJUm
lSpWSj4+PIiIi1KxZM0nS6dOntXvXsOnXfvvXo0cKyZM0nS6dOntXfvXo0cOdKy7AAAeKJnv3nb6gguvn9uuNURAAAAgHuytFi7cuWWKfvvtN+fHR48eVXa0
e1c6dO5UxY0blyJFDPXv21PPw5UvXz7ly5dPw4cPl7+/vFl7+/v1q1aiVJCgoKUocOHHdSnTx9lypRJGTNmVN0aFH
nLqEAAAAAADAo2BpsbZt2zZVq1bN+fG9tNB2jTp2mT5+ufv366dq1a3rjTd04cIFIFlStXTitWrFBAQIDzc0a
NGiVvb281a9ZM1e0dU40aNTR9+nSlSpUqUqxb8fAAAAAAA/HtYWqxVrVpgk73c4HBoyZIiGDBmS5Dl+fn4aO3asa
sxo4d+wgSAgDgvrpLn7U6goavljb63OgIAAADwWLHtGmcyfgQwAIAAABSkG2LtfDwcIWGhioIkIlSpSQJN28eVORkZH68MMPJUm
iijUAAAAAAADADRRrAAAAAAAAgBso1gAAAAAAAAA3UKwBAAAAAAAAbqBYAwAAAAAAANxAsQYAAAAAAC4gWINAAA
AAAAAcIO31QEAAClvzNw6Vkdw0aPVj1ZHAAAAAIAHxog1AAAAAAAAwA0UawAAAAAAAGAAAAAHADxRoAAAAAAADgBtZYA2A
7i6fWszqCiybtf7A6AgAAADAhhixBgAAAAAALiBYg0AAAAAAABwA8UaAAAAAAA4AbWWAMe0M6JDa2O4FS887f
3PWfN5PopkCT5qr26zOoIAAAAAAA8FIxYAwAAAAAANxAsQYAAAAAAAC4gWINAAAAAAAAcAPFGgAAAAAAOAGijU
AAAAAAADADRRrAAAAAAAAgBso1gAAAAAAAAA3UKwBAAAAAAAAbvC2OgD+3U6Na291BBfZu0617oIgAAAAAAPAQjg
DAAAAAAAA3ECxBgAAAAAAAAAAABWA8UaAAAAAAAA4AaKNQAAAAAAAMANFGWAAAAAAAAAA4AaKNQAAAAAAAAMANFGsAAAAAAAACGyjWAAAAAAAAADdQrAEAAAAAAABuoFgDAAAAAAAAAAA
AADdDrAEAAAAAABuoFgDAAAAAAAA3ECxBgAAAAAALiBYg0AAAAAAABwA8UaAAAAAAAAADfYulgbMmSIHA6Hy+3W+/ejADFj1ZHAAAAAIAHxog1AAAAAAAAwA0UawAAAAAAAGAAAAAHADxRoAAAAAAADgBtZYA2A
sXaxJUuHChXX69+ePc77qqnbFRoaqlq1Ro4cCPb0KqlEXzbc8zDOVj/VV69+VGvKL+//rtszAxAAAAAAAMC/ge2l
vWbJkfTPaLPXMBR1v2+SKmgUCUqyaVW1cFq05d+BP2tmJfZkhT3fkn03/5nkjFHkBtmIlvl+Y8uH2dszLhDM3UCnaXAz/ZZhrwXb/
/+VRWSj4+PIiIi1KxZM0nS6dOntXvXsOnXfvvXo0cKyZM0nS6dOntXvXsOnXfvvXo0cKyZM0nS6dOntXfvXo0cOdKy7AAAeKJnv3nb6gguvn9uuNURAAAAgHuytFi7cuWWKfvvtN+fHR48eVXa0
RVPKJ5ZKGaJO9hBEKzzLBJ8kUnI/OVa7EpkCR5kpM3xkZ5peRlvnrtVgokSb7kZL5mo8zJyXv9qn3ySsnLfMMDM9+
66ll/f7FXb9zz/pSWnJ9x7FV7PfclL7O9rjGSl/lqCiRJHk/LKyU3c0wKJEm+5GW+kgJJku/+j3GXUyhJ8iTv+dp
umf3ve85V22X2ve85Mdfskzk6OvV9z7lio7ySFB3td99zLl+3l9zL19+CqKZZ0Wnve87l6/Z+Z8yNm8j59t1cF2Sn/VGPsaThsFANmS1QMsA/y1HXmsrEXhm4hVtwqrBOoabOOH1RLEpxmw
P13d2GPGP0caY+36ewyTnLBv7888/9cQTT2jjxo2o2qWLGi8/jw4cM1Y8YMHTnDBkyREOHDk3JmAAAAAAAPA
gJ0+eVPbs2e95jsePwdHcbGJDgWb8CAAerdu7fz49u3b3b+v8+fPPKlClTkp/jjjuju2oaIWFhenkyZYMKDAx8aF/
3USLzo+dpeSUypxRPy+xeSUypwRPyyuROaV4WWmZPyuuROSV4Wl6JzCnB0/JKZE4JnpZXInNK8LS8EplTgqfllcicEjwtr0TmlOBpeSUypwRSmwd3PmzZ1L7V19Wt7mz
Z7nuuxxdrmTF9u3bcv27fmQtlirx0GnZZbpdULiuLLCTrZaxezkNRsu5cOG/nt7SZXNhZ5Nw0hlckYc0mbaRFzXvNHUM6NNtrNUkK9NksG2FGbY0o
5JTKnBE/LK5E5pXhahZk/LK5E5JXhaxonMKcHT8kpkT1VqlQpRUREk6yt6zSoxFNt1sJL0uKYsvTjO6x52kapyIXpzK8YaYgUsWh7wjGhN+HLcwlsTkc
wmRoKAAAAAAAPEweP2JNknr37q1p06ZpypQpMm7lq1T3ZSKXpYgMTzaoq2IeW4ajhZyzkdnvUrNHnhmjk3IpmFTSdzlRx4m6H+F1A1iLOm2ofCKbfSt
wYcN0+vRpFSlSRMuXL7/vhgUsWHA6tyOQqNoqmgiqqQjqSinFzvzGeTvsw+cGsfZCwiAArdexIzWpxqRPUSN9M76cN1f0d4Bwlp5jHqxqkJvhcRPa3pZQC2e8kZVHxg1+VKw1A1Y49GrVmkJ8sC
B0/JKZE4JnpZXInNK8LS8EplTgqfllcicEjwtr0TmlOBpeSUyp+xBgAAAAAAFiBYg0AAAAAAABwA8UaAAAAAAAA4AaKNQAAAAAAAAMANFGsAAAAAAACGyjWAAAAAAAA

KNQAAAAAAAMANFGsAAAAAcJfY2Fjlzp1b+/fvtzpKsty6dUtDhw7VyZMnrY7yQDw1NwDEY1dQeJxz585p0KBBWrN
mjc6cOaPbt2+73H/+/HmLkj1+bt26pbVr1+rIkSNq1aqVAgIC9OeffyowMFDp0qWzOt5j4fbt2/rtt98S/V1+5pl
nLEoFwE5iY2P12muvaeDAgcqdO7fVcWBTV65cSfA8EhgYaFGax8cTTzyhlStXqmDBglZHSZZ06dJp7969ypUrl9V
RHoin5gYAiWLtf7Z06dJkn9uoUaNHmMR9GTJkkMPhSHDc4XDIz89PefPmVdu2bdWuXTsL0iVUr149HTlyRB06dFB
ISEiC7G3atLEo2ePl+PHjqlu3rk6cOKEbN27o0KFDyp07t3r27Knr169r0qRJVkdUxowZdejQIWXOnDnJ3+N4dix
cN2/erFatWun48eO6+6HY4XAoLi7OomSPr2vXrik2NtblGC883ePpf3+eJn369Prll19sX6w1bdpU06dPV2BgoJo
2bXrPc7/++usUSvX4Onr0qLp27aq1a9fq+vXrzuPGGJ5HHpIPPvhAv/76qyZPnixvb2+r49zXc889p+eee05t27a
1OsoD8cTct27dkp+fn3bu3KkiRYpYHcctXBc9Glu2bNHatWsTfeP8008/tSiVZytRosQ9rzXv9MsvvzziNAnZ/9n
B5p577rlknWfni5tBgwbp/fffV7169VS2bFkZY7R161YtX75cXbp00dGjR9W5c2fdunVLHTt2tDquNmzYoA0bNui
pp56yOsp9LV26VPXq1ZOPj899S1i7Fa89evRQ6dKltWvXLmXKlMl5vEmTJnr14tWqVKkEee12vRnPE3N7e3srZ86ctryOuBd
Puy7Kli2bqlatqqpVq6pKlSoqUKCA1ZHuafjw4frPf/6jAgUKJHhctttj9GeffabXXntNf75+930etPq5787e5fr
165owYYIKFSqkChUqSPpnwMK+ffv0xhtvWJKPEWvQ888/r1q1aun11193Of75559rxYoVWrRokcaOHasvvvhCe/b
bssSjl/ylTpozGjh2r8uXLWx3lvry8vBQVFaXg4GB5eSW9pKEdi9fMmTNr48aNKlCggARr1y7lzp1bx44dU6F
ChXT16lWrI3q8tGnTateuXcqbN6/VUR5IYr/Ld14o2O13uUuXLlqzZo2GDRumV155RePHj9888YHjzcff/yhzz//XB988IH
zhamdMDL30Rg6dKjefPNN+fv7a+jQofc8d/DgwSmUKnnef/99ffzxx6pRo0aiLzqtvuD1dOHh4S4fnz17VlevXlX
69OklSRcvXpS/v7+Cg4P1+++qcjISK1du1aHDh1SSEiIqlSp4iza7DZNOyQkRB9++KFHjL4MDw/Xtm3blCl
TpgTPg3dyOBy2eu579dVXlTVrVr377rsuxwcPHqyTJ09q6tSpKR/K4KGJiYmxOoJb0qZNaw4fPpzg+OHDh03atGm
NMcb89ttvxt/fP6WjJWrLli2mevXqZu3atebvv/82ly5dcrnh4ciQIYPZt2+fMcaYdOnSmTNnzpjIyZAhg9m3bp2
2MlqivLy8zF9//ZXg+N9//228vLwsSHR/1apVMz/88IPVMR7YxYsXXW5nz541K1asMOXKlTMxwcPHqyTJkyeP
aNcYYYwICApyPdzNnzjT16tWzMFnS0qVLZ3bu3Gl1jGTzxL8/T5MrV64kb+Hh4VbHe6zMmTPHVKpUyPHVKpUyfz666/OY7/
++qt5+umnezZsy1MllDVqlVNRESE1TGAf63ixYubdOnSmdSpU5v8+fOObEiVKuNzsyBOvi+JFRUWZL7/80rRu3dp
4e3vb8hojNDTUHDp0yOoYj7XAwMBEf8aHDh0ygYGBFiQyhqmgD1H69OlVunRpZ3teuXLlBO8o21HGjBn17bffqle
vXi7Hv/32W+c7LzExMc4pP1ZLnz69Ll26nDvqlVLo0eP1hdffCHHpn3cqrly5osGDB+vZZ5+9ZZ5+1OF1
CJonBtzdu3JCvr28Kp0na7t27nf/u1q2b+vTpo6ioKBUtWlQ+Pj4u5xYrViyl4yVLYtO7atWqpdSpU6tXr17avn2
7BamSdv78eee7cIGBgc4pRpUrV1bnzp2tjJakJ598UteuXbM6MrJ5yt9fYm7evJnodNscOXJYlChxR48etTqCWxY
uXKj58+frxAgUKa2pRpUrV1bnzp2tjJakJ598UteuXbM6MrJ5yt9fYm7evJnodNscOXJYlChxR48etTqCWxY
uXKj58+frxIkTI11udjN//nxTqFAhM23aNLNt2zaza9cul5tdBQUFORdCDgoKMvv37zfGGLN582ZToEABK6M

lKlu2bM6MeDQ89Wd87Ngxl9uJEyfMtWvXrI6VpBkzZpjr168nOH7jxg0zY8YMCxLdX5o0aVwWTsfDV7p0abN+/Xq
rYyRbmTJlzMCBA40x//ei8/Lly6ZRo0ZmwoQJFqdLXHh4uNm+fbsx5p+f96RJk4wxxvz4448e8YLz4MGDZsmSJea
bb74xBw8etDpOoho0aGAWLlxodYwHljZtWmfp+sQTT5iff/7ZGGPM77//7tw4y+6OHDliAgICrI6RgKddb8bz8/N
z/k7cWazt27fP1r8TAwcONFmzZjUfffSR8fPzM++++67p0KGDyZQpkxkzZozV8R5L27ZtM23btrXt5gU5cuTwqDf
u4mXNmtW2z3WegDXWHqLQ0FAdOHBAJ06c0IkTJ3Tq1CmFh4e7rDljR5s3b1arVq10/PjxBAtQ22nKUbzmzZtL8rx
1qXx8fJxDakNCQnTixAkVLFhQQUFBOnHihMXpEurWrZs+/PBDTZ48Wd7envFQ4WnTpDzxZxwbG6u2bdvq888/V/7
8+a2Okyzt2rVT3bp1FRwc7HL88uXLateunV555RWLkiWtdOnSOnnypEdNefeEv7/o6Gjnvz/88EP169dPw4cPT3T
zELusYxfvwIEDzvUtvb29de3aNaVLl07Dhg1T48aNbbngdPXq1fXtt9+qZMmS6tChg3r16qWFCxdq27Ztatq0qdX
x7it//vy2f5xr2LChevXqpT179iT6e9yoUSOLkt2bp63xlJiFCxc6N/myE0+73oxXuHBhrV+/PsH05QULFqhEiRI
Wpbq/OXPm6L///a/q16+voUOHqmXLlsqTJ4+KFSumzZs323ZNu1WrViW5ltbUqVMtSpW4HTt2aO3atVq7dq3Wr1+
vy5cv66mnnlKPHj1UrVo1q+MlMGTIEA0ePFjTpk2Tv7+/qZMmS6tChg3r16aFEmXx8bHtmwXx4l8M7dq1S4ULF3YZFRgXF6ejR4+qbt26mj9/voU
pE+fl5ZXgmJ1H5nrK319kZGSyz7XLTtjxnnvuOdWvX18dO3ZUv379tHjxYrVt21Zff/21MmTIoJUrV1od8bFxv6L
YTjujPS6OHz+u7du3K0+ePLa7Hho6dKjLx15eXsqSJYuqVq1qy12PPe16804//vijhg8f7rJp1qBBg1S7dm2royW
pQIECmjlzpsqVK6enn35a9evXvXV//+/fXVV1+pW7duOnPmjNURE8iaNatGjhypl19+2eooyRIdHW3rIu1udz9m3M1
OO8ne77XInez0uiRPnjz67LPPVL9+fQUEBGjnzp3OY5s3b9bcuXNTPBPF2kO0ePFi5zDVffv2KVOmTHr66adadVtWp
VVatWzfnOst1Ur15d/fr1U926da2O8kD279+f6DvJdp32gEfrzgfVDBkyaMOGDSpcuLB27dqlxo0b69ixY1ZHfGx
069ZNM2fOVN68eVW6dOkEI+3sMtog/sJm6NCh6tOnj8u0/PgdK59//nn5+vpaFTFJ99s91m47/fH39+j9/vvvunL
liooVK6arV6+qb9++zhfLo0aNst3vRLyLFy9qy5YtiU43suM0bMlzimIA1uvfv78CAwP19ttva+HChrWZsqVy5cq
lEydOqFevXrZ8EzJTpkzasmWL8uTJY3WUB7J9+3YdOHBADodDBQsWtNWIeKSstGnT6sCBA8qRI4eyZs2qqZcuWqWT
Jkvr9999VokQJ58jRlMRU0IeoU6dOeuaZZ9SxY0dVrVpVrZGRkom822nV9J+mfkYGjRo1yvlB+8skn1bNnT9WsWdPqAn
ExcVp8eLFLi/qGzdu7DHro3qSmzdvJlrE58iRw6JE93ZncfbCCy8oe/bs2fJETVe9257swZtWjRQmvXrlX69OlljNGlS5dUrVo1zZs3z7avTfDoZM+eXadPn1aOHDmwOb5gwQJDb5ouXbpZW4GDhxosmbNaj766CPj3Yt5sumjdvnvNnmip
VKjVv3lzjx4+33VpPMTExzpFJ8Rc3qVKl0iuvvKKKxY8fabjHO8uXLa+PGjSgSZMmZtWjRQmvlypceGx6O9OnTmyNHjhhjjMmdO7dZvXq1MCaY3377
7zaRJk8bKaPfk7+9v1q9fn+D4unXrjL+/vwWJ7i1Tpkxmz1CY6vXr3aZM6cOeUDPYYKFChg5s6da4wxJl26dM7
f64EDB5ouXbpYGe2eqlataqpVq5bkzW4GDhxosmbNaj766CPj3Yt5+dn3n33XdOhQweTKVMmM2bMGKvjJapBgwamceP
G5syZMyZdunRm//79Zv369Pj5+dn3n33XdOhQweTKVMmM2bMGKvjJapBgwamceP

3IigoyOzfv98YY8zmzZtNgQIFrIx2T+nSpTOHDx9OcPzQoUMmbdq0FiRKmp+fn9m7d2+C43v27DF+fn4WJLq3dOn
SmR07dlgd44H5+/ubAwcOWB3jgTkcDuPl5WUcDofLLf6Yl5eXeeaZZ8z58+etjuqxkvqd/uWXX0xAQEDKB7qP3Ll
zm++++84Y80/23377zRjzz3N3y5YtrYyGRyjhtmNwW79+/bR69WpNmDBBqVOn1uTJkzV06FzK4LTPr5kzV06FBly5ZNM2fOtDrePR0
5ckTdunVTzZo1VatWLXXv3l1HjhyxOlaiihQpot27d0uSypUJ2rjxo0aNmyYrd8Vz5QpU6LTPYOCgpzvgNr
J1atXFRISkuB4cHCWrl69akGie2vXrp1WrVqVYMqDnZ04cUIVJk0aXL1+WJL388su2G9Z9Z+p+LFi+upp55
y3goVKqSbN2/ql19+sd2UGEmaM2eO/vvv/6pv377y9vZWy5YtNXnyZA0aNEibN2+2Ol6ifvrpmSSRl5e
XvLy8VLLyZY0YMcJW73Te7ebNmzp16p16pRoNnDjhcrOjFi1aqHv37lqzzZ8o3i4uIUFxen1atXq0ePHmrRooXV8RKI
1rhuycuVK5zuyTz75pE6fPm1ltCTCTVr19fb775poYMGaJFixZp6poYMGaJFixZp6dKlLje7yp07Vt3N92fj17Hbv3q2vv/7adqMvfXx
8nCN7QkJCnH9vQUFBtv3bk/2eFIxEtakChpBQoU0F9//ZXg+JkzZ5Q3b14LEt1bWFiYR10LxxSt
UqJDtplonR0REhMqUKaOIiAhdunRJly5dUkREhMqWLavvvvvvtO69at07lz59S3b1+rozq1b9/eec15p5iYGJelgey
ievXq6tGjh8tI1j/++EO9evVSjRo1LEyWuPh1yyyUpXbp0zoX0GzRooX0GzRooGXLllkZ7b4b4uXryoyZMna8CAAc4ZHb/88ov
++OMPi5MldPDgQXXXt2lU1atRQzZo1zR2fONPXq1bMw2b0tX77c+Pr
6mrJly5pevXqZnj17Xnnj/XqZnj17rJly5rUqVObFStWB0vgeXLl5tFixYZY/55V7lgwYLG4XCYzJkm1WrVlmcLmff/65qVm
zpvnzzzz+dx06fPm1q165tJk2aZGGyxFWvXt28+OKm28eNSeGdqbDe4UkcYQeRVWKlcyJEyesjudi69atzuL3y7tzuL9z
Pj4/JnTu3yZMnlb8zDzxs1zzOxhMMJlYHtN9/Of/5iXX37Z4nsuLC14RJRsWJF88wzzj5jvv//e7Nid0hHx408XvZO/du88QTT5hMmQ
xxBPm559/NsYY8/vvv9tuWPudihcvbt56Ex9966y1TokoQJCxI9uHPnznbnt29bHSOB4sWLmxIlSjhv6dKlMz4
+PiZPnjzO+J4V06dLZ8ue8e/du88QTT5hMmTKZJ554ItEpEXZw4cIF8/nnn5sqVaoYLy8+TWwz5jvudi69atzuL9zJk
zpl69eiYgIMCUKFHC1n97uXLlStbWnJ5duZcZHZUgWpu6dI+x/3AH+nqk8+FCzWalKZ9TLDKGPGcfYcfefP8M6
fo6e8QWPMP1OF9+zZk+D47t27nVOFjx07tos3xS5dumQXQxpzKyS6+FMmSz+ZPtU5tWrDC
fffaZGTNmjImIiMi+LA6Tp1eeust8/777xj/q8Ayps3r/H19Nbdd1KhRw5tGsVRy7Vv7Z55pvGFmc3c32d8t51L9zjy/LRnzx7l5
P5fihQ4dUrFgxXb9+3aJkCd26auXOnihQpYnWc+/L0BSGvXmbRoUKFVLr1q2VJk0aZ6ve6P
d16lTp/Tll19q6tSpOnz4sG7dumVV1pAru376t27dvvy9v7nw2a58+f71zw/fXXX5evr6/FCR/MrFmz9NZbb9lq0Wl
J6t+/vwIDA/X2229r4cKFatmypBvlXLlyuXcoOeiDOmICP/74o2JiYS0aVP/vvvvatCggX799lqx/1lap/2a7
Xr251RBdp06bV9u3b9eSTT1qVdVxe7Tdzr2mRc0h7/CouLk7dEuywcwmEOmyVhckTWEiUV6YOYm6XaVGQ5Mz
rQx5A+7Pl7yLrSTsV1hRb3/Ws7/bjGOvm0u0CIS3nb8nqTA7ua4Eu5nfYvmw/U+5MH7yDDw0ICsvpkkPeEEd4pKBHSVSaVsl5+U9pmdnOsWLFqJm
j1KOJBwWJQuEpUqoRLdiilp/5lq8uWgEnH9ibzoplYJIV3JYCM9n1S6RnTJuv8ioqeduhfwRQx/QFH0hmRRfv7vGm+IMUXWXzKC5CUAXzQc3TeJ
kRFAKTBSaOqnnUG9ZYbC2eHRU0gHQ1+Lk3zzoYMVnmxOOeCW93PDZtVy1MaBdJD4O5tKbTBWNk2lKCUiRbhBuAKR+Kl/vA9QCv+hD6YQXqz6K
K03pVQfr3Rh41EoZbSuShdSExUdmWutmrOSwOk3zsSUzaRt+Tu6a0W8GxhHwvWkMIxJjRvq5mqPt1gFkG+zMoh1EgfJ1fPK/kFB8tRpc1UkzlP
6bmL+PsXetm6LqfabZbpJjH8WYJFPYQwbcO07S/u9EZr+67pdlSqsqYO/WYSVDWDa8jdtyCWk/z0Y+5bKYbM4/ux6kEckrBmEX/2O9DvibVhO2
QiNLIkL4PcRw7Y73IuxGDqY6g5YmjsC0lzivSjoeqbG7H7A7ghhdwlTmd1NpQ5Mdcs3QMbdtYS/9dHptbA1vPfDO7cmKZRHLtLXFJhJ4SrOsLc
nNltlbdp1Lc6pkqkIDQuWsV7VvLsptvEjCzMkCwa6L8hZ7zRSMtQBwUN9vUVOSCvz+Ww4mLL4QwCM4qgRD0VGdYHB8UpqVdiDvrb5dCaxuJn2iI
h4ZRPmDiFrvvuqX9ax7YAPQL6Xqv7H9BTYo7fJhX0aKyXrDsM+pE7kGmhT6QI0PpTRftKz+jRevLzJmcD5S9ZvjQUKKtStUPd8B0j8hRlOiGPE
YpY+z9ji7qlk1oYW4BKyeIJvK3Pwwtyes6+hY9m3hIzkW7cYgIXVfJvZwRJO0FrMUcvSpTcCbUkhwvZWNmdm5qvaGiqNsD3kKOv2GJh/FWz1Jfi
jJqe/67UuS+Mft/lmtKNG9GVVW7tKmbNcuYGnOUFncnwCamJKCE2yYY/2VTmoNYlCsFWWFKgimyc9b7/+XhqalF63+GR29bgxXKO5UWnnMd5e
Dp7/+R67vlNLZfh9f+ffZxyYKVqBd/YxYZaAO06YP5DRqwTajFkCNUqY7r5UhSu9ekkX5+CbNSVcP1MSkIJyLkKbjUUbYkWAe93bMqOjqSrvOY
tyRUsbwcUmpQUmv1yoRZhNEBxdNjwaDXzdrXCQJm/1i/BYotKQsK+Eyf9sIcjCrTz7MmwwgYAT+PvjdCEMpyPPX1fGUn1tBOePPTY1eHDM4yRs
SkKP8sDNQqqzspSatXeOp2YfAbXEuUGHHlmaYmfQ7YzX7lKWZ/q27eO0sXYxzWHTqfpjqUNpt3G2KAh3gU/rh3qDWlhyqWcGXzrJxC7J2uwpmK
SdkJVOwZaydpsaYqtr9n/ltG9e4xs1QJdFvdYzYlqDMtKJ/wf7l9+j6ycSL19ZlJ2MdamrtbPptLIdd5xjBlM+8JdVpVQOMYpy3ZLaBKnrr8U1M
H9n/O/6XFfT6FCTXjpZjS1DiUJQ/mT/ZbWVvt+6ej7X3N7MDtzsMe79FPhM23ucyxX2KtsfeDBwb9NTC+rjTVSe0GvpX4bRgGiR6sS0OMHIEE4HS
95lpa6eqmQJmYm5RHvKaSSSEJp1NcVSj+GS3RZXyRVOPKbybzNJo+tLO4lS57PRV12xmcqK9VTYZLDoWQ2MTxlPZ1S7KX5dSMM8w9ruPCpL3qi
X4BDvj6GNkLDZjB0K/rSmRy5OBUdm1JDmNnlrk9Dr1MCptOJMlFlz7VZWvHP2ZZHmpbIPVPF3NGzufNaZhlbVYVq33KtWzK1mLUslc3TerhrsZz
71Xa/zMXh37TMk+dsj+pDB2GbYnO+Bc79bB1t

uQ2mNMXI4HLa7+PXy8tJff/2V4B2KQ4cOqXTp0om+o2Gl7Nmz69y5c6pTp45at26thg0bys/Pz+pY93S/Ifk+Pj5
q3ry5Pv/8c9t/L3aWNWtWbdy4McGuwYsWLXJO1bCDu0cC3svXX3/9CJMkz92/v/GPZXey6+ObJIWHh2vo0KF65ZV
XXI7PmDFDQ4YM0dGjRy1K5ir+RX3x4sW1evVql4I1Li5Oy5cv1+eff65jx45ZlDBpWbNm1ciRI21RlDyIJUuWaPj
w4Ro/frxKlSolh8Ohbdu2qVu3bnrrrbds8yJZkurVq6cTJ06oa9euib7gbNy4sUXJ7i2pYnv8+PF67733LC+2vby
8FBUVpeDgYOdjXWIvmez4+J4Yv9Xz41b95c7733nvNYTEyM6tatK0m2LeILFCigSZMmqVq1ai7HIyMj9dprr1m7299
j6urVqzpx4Sn4SCN23sVhZ72nNJu3btVLRoUfXu3dvqKA8kOjpazz77rPbt26fLLy8rW7ZsioqKUoUKFFfT9998neM2
d0pJbBFv1uMyItYcoV65cKliwoJYu59XXJAo8efI4j589e9e1bh4eG2e+KN53A41KtXL/Xq1cu59XJAQIDFqZJmpwvaB9G
6dWtJ/wxXDgkJsV1hGS/+KbtD4uDjt3r1bFSWtWtCpekgYNGqQXX3wxTPo1u8eLHeeustvfnmmyp
btqyMMdq6das++eQxZY3WE/8np06cTfSyrWLGiTp8+bLvvvvatOmTSpVqlSCQsqOvxPSP7MLNmzYoDVr1mj79u26ffu
2SpYsqZo1a1odh6q1h6xgwYIqW7as+RWtjvJAxo4dkvE8+wVsdWkWZWxWSa1ySNcAQXavE8dd2yxdD2eNOmTbM6wgPx9LVx8ubNq/nz5+vtt992Of7VV18
lWLPDSkePHH68ccfVbVqXVl5eWnevHlKnTq1bibZvmIk3sJDg7W7W7t27lStXLpfju3btUqQbgkLC/OYa8w
//vwjTvNgBg0apHPnzqlmzZ2pav369li9frldffVWzZs2yxdD2eNOmTbM6wgPx9LVx8ubNq/nz5+vtt992Of7VV18
lWLPDSkePHpUxRrlz59aWLVtcRhH7+voqOD
zZ6L/9gTh4eH68ccfVbVqXVl5eWnevHlKnTq1li1bZvmIk3sJDg7W7W7t27lStXLpfju3btUqQZMmawJlQw7duzQs88
+q6tXryomJkYZM2bU33/9L9ZEuyKEHgOkXfL3uvvvvddEfmJ9KDnCnKj7gjsMlNoDnhv4EjTkkgPuu3m3ikuUvjq8Nyuxu62AStVq6tJq6lpTjP/e8euZ9i6ElOinL5QLbcsQ
kyZOVPn16bd++Xdu3b3e5z+Fw6O3vbz4e5z+Fw2PJ34vbt25o+fbq+/vprHTt2TA6HQ+Hh4Ksolh7iBwOhyZMmKA5c+a
ofv36GjlypPMPx26/jCVKlEh2JrtOrfQ0ZcqU0cmTJ21frMW/uM+M+VK5f69u1r64uwu23dulULFixI9B1DO0ydu9u
ePXsSvWDPmTOn9uzzI+mf0Sp2eJHqhSS86440ZM0Yvv/yyi/bNmS/Pvzu3Pzb/PmzbVVu3Tp
VqlTJub7oypUrtWDAqvjOcU/Rtj9ndrEXL9+XVWL9+XV988YXlygrbPGlaWK
hZ/cqtu/eTMsu9u/fn+jjjmx03wSlSpIi+++471axZUsXq+8TlEgTndy3rxM+c13ezR7Xhq+vdysgIMC5QVMuEmiWK1+f4yQdcuJ0MtNWqFvLUd8vEz2p
aWJwuab169VLDhg01epU+fXps9eeukNs5Ul5ZeE9gU+W
GqH79+lZHdOFJzyWS541yNcaoaNGiyjBzU3VsdMwVdvMVnGG4jRVlj7SG6cz2
GH374QS1bttQLL7ygQYMWmMs6meu3WL3W5fKTj/nOx05ckSvnL9ngi+E8qbXVRsTEzgB40c4aD8ncaqSfN2rYaEmzbr4X+QkWQVwG8gBoKtW6xNd5syD8bU5jbSnbwoTPiSuyU8ptdK2ZQk6ppEqXiCMGwwuMFo4fvmvMXLSkIbHe+nQcJXT2kzvg
GH374QS1bttQLL7ygQYMWzMs6me3EWL3W5fKTj/nOx05ckSvv/66XrpzzfPCz/ZzLV9R5s99HQcNUeyX06xO+Adi4kaNp8gbPTmM3j5P35fl4uhzw/
R7dq1FRERodq1a+/vw4cOKiopSkyZZNbDkaqESJEnrqqaf0xXVafdpRfyNfXV9I/Hof6PxdOOFB0acSVbqcqJ2u04/jeUabC5w0Zq4LgL9r3pK4tUTWvVK
kbbU7FziNFxsbq169eL27do9eq3evD1Hk+KIoPDz8no9zv//TYg4NPsaVK9cvXKFlk/iCcchtq8MzwGnuT2+
UqJBee+019evXT5s3b73b7Y462CFMWv6lFL1s0D6bnJ0qwWpx+QjqO1TN1s5Ad3lZDPTDfrowqYjIJnhOWc2j7Zfyvm6hQWKAovwffyM3M4evBrLNjFcepQyYT
duiV/f/8E1xZ2G0Ecb9GiRWrevLlq1qzpUmyvWrVK6/qa/GP0y9fAnTr/eDyrGpjOFVKpkPFsbUoSGfnCD67zg/xiCfAZpiYOsZzs99PBnHB8DDL8pmFaJ5U9XcdIHw9s5LUhqnrsh
jx48rODjYpVSz6xvmN2/e1MvuwFEuKS
+/vlntWnTR/++qVERMIDAx0jg7MlSuX5syyZo0qVKunoFK9FFq0Mz4H07W2aG2euH4GrWsWh7ctx8Gq6q9Mjrd
LEvysV69erreee07jxo1Ls+LsCatle43UtSK62RGrD0i9erVO6ZNm9SoUSfzvR1HktcCMS9cG8LJFLy5PuPYkZzrEoRvMf9LkzXKiatNk0kpcnl88EDtMqTqUlqQmX9U1Fc9wM9O8SIXUbpIqc+QRgY5Tmaw2a2NO5
QjBkzLNsFJDnOnj2rI0eqF27ds5j8Rdodlz8VpIWLlyo+fPnJ/pOgN0uzIY
OnTo519Wym/Hjx27UGzk27dd+EIkbw/ihUrJofDod27s4esA/NSbMVX5ycP+VVmz75S1bg4TlS
nz54uH8c/zi1fvlxvvvmmNaHQwcOFCZMmVSu3btVKtVK73DSh7Xfbmq+vEaj9CpdDNZRN3S+YJYNy7LZPu+wjqYSDDhkyLaOv
LnZiYGL311luaP3++bRjqqYdPSvqHu3VryWUMY7tALGmi6kyGZOSaHQNXdi1h43B/FfrQXcXrDfQvrqS3mcDP9dagZEYaaR
r5cqVzinZ586ds8Uyhst7hbs9DOOHaXTZZQdWKI7p2uMeHqRHwVX+dQVuYuxeyWPBl+CKWusXGizzmWF7eUdLK3wnM1aUcfW5tQPUHwqa2wcZjh21oK0CCB
brg1XoEABHTx4ULly5VLXmbx4gCzUsSlljtsOsUdKgwm69I9Qm9hZcZ9PInXjX5p9N5QNdZ7tQK/Vz4PQGssbUFHzA47V
rf//+mjNnjq2KNVuOFDV4aKpWrWouXKz9b5MsWYopa8sR8sp9MsP3QMKxxO1nHLiq1ekeDJtIjWFjiV1hhOxG4m6ojqZ/0imt9n8GmLnQ
ePWqOHTvmcrObMWPGmHTp0pkulXbYX19f06lTJ0OzZk0TFBRk3n77avjJeDv72+OHj1qjDEmU6ZVfu3cYYYY/b
v329CQ0MtTHZvly9fNhMnTjS9NJmmTTLR0dFWwx7qnKpuBC5y+qgsWbIYLy+vBDc4O8OuPGjjTNt27a1
CoaGh5ueffzbGBMQEGAOHjb0kDhQpYxZGm9PPxzRKHuXt4ilZhx4TTMQ4ifgzpHMxuq2pWF5pxoTEbilWVj8KJ2bN5H
8+LBZvny5uXr1qjHNu3b1uc6N7eeOMNUQr7BgQbNgwQKTJ0aM3XqVPPuu+mz9mzZ1sdDvXxU6ZDbxZeuX
YYwIDA82vv/5qjDFm1apVSW4pZ3ajlMvEvkSMLJDjnb3Z7v4vVWhgY2UnNRakLZsmVvl4zdyPZTpnGjz8
jjPnll1+c13zMmEpcdHrAcTKHxhLyXQx6XQbGqVZH0Yv5a71+c13N+fn5m3rx51oZLxMqVK6M1
+vcmQIYPZt2+f1VGNMcaEhISYHTt2Jhn/L7/8YkJClIuUDIEBQU5H4uDoLMvo3j/F37zUjTHGbN682N8RQoUMCSTIxYe4g
Sa9MzZsyoyMhIC9LcmycvRJ6UcuXK2XKXXVzaVZz4ZzX2XrnXKqZM2aoDMaD8n2koN9y+3lXkTX3pmNh0tWbLD63YT1oWaOsNZP3qKPj5kfaMkbUGaCG+eVT5Xtc3kuBUjk/
L34mMGTM6d7V94okndOfOHWXJkkVHjhyxKNnDObL0rkVXv/Zh6ytL/SjF8Rod6aPLkU3eDa3BTEFHMjvR95OhWQ2ItM6IOxhZHX0xoJr6Rvh6oQbZN2M57BsfcKRdKhr7vHJBgHOTLoNgd9Aya
+uT744AOr4z3W6tWrpwEDBkiS8vPzbU6eP39+c/XqOdzfZ9Znt2u0ubCZFV8X9Hl4Vu9+NPbx3ZYsz/+h6xXxjxMH0mNNxZTmJbr9+mXMlqKFTAKVLlkvRuGZOm
uXFHdunV19uxZrV27xBzDXSm5X+n5ju4HC75l8g7dRcdgbN24GrNq6kHTW4Qw20bNvdhNxxmaXb6Y1SN+S+fS+nP+d5pQgAl6tNJz+Nsu6qO3kZCc2xwZ0
p09tuNGO8b7/9VjNnzlTVXZXvn17Pf3008nq7JZDvbQqrezTTbKNuNRGrLX+uGZYVbW1pL6Wz17Ot6wsF5z88fmVZ2CXyCknV+tiqeJeIrwKoT
t2/rtt9905syZBGsIxIxMdZ6mnZ9XIyMvcWD6uGtcWZ79Cy29HtEgbpFVj0hOjFc0Vm2Sr+BxTYVa2VgLjrMvOGnkz0AJ2bGPB8oGNkwGnqy0

oJT2yJzF2W0cr3vDhw53Xn++++67atGmjzp07K2/evLZ7rr569arefPNNffPNN4qNjdWKFSv02Wef6dixY/r111+
VI0cOZc6c2eqYCTz//POKiIhQmTJlXI6PHj1agwYNUteuXS1Klrj40WjTpk1zzuKIjo5Whw4dVLlyZXXs2FGtWrV
Sr1699OOPP1qc9p/X+iEhIUneHxISogsXLqRgovuz40hRirX/UXR0tMsfzL3YYXpUPE9fiPxu165d09ixY5U9e3a
roySpevXq2rVrl8cUaydOnHDu2JUmTRRnRUP8AvDjxo2zMl4CTz/9tCIiIlS0aFE1a9ZMPXr00OrVqxUREeGyQ6/
VPH3NJMkzXnR+9tlnyT7XjrszJWXhwoW2mTpwJ0+bVvvLqq6/q/Pnnzeu6557RkyRINHDhQUVFRioyMVLZs2ayOl4C
Pj4/27t3rcVP+evXXqJR8fH+cFb7zmzZZurV69eti3Wzp8/r/DwcEn/XLvFv5uXqJJldTXVrcpoCZgklkq+ceOGbde
kkqTNmzerVatWOn78eILvsW8d+ZEPlpenHEwTR+8SqE7H4X5c8HYt9xAyKLKzRyyK/x8oqRdaA+7D3RtEfSNwoBJCdo
RQWwG6HmkV4ZPVVyZfhxmRJjyNwDXb7ZbDNXpfo6+wWdYtrJJ6zqvf0m7hq0kPSQXpQ2m+MNsfEsxXlzZa1b+nwTJ
Y+x9lyJDB+U5h+vTpE33gNzZcQ8tTd7+SEo62M8bo8uXL8vf3d1krx24aNmyJCBYbtVqKEhob
q3LlzypkzLmzKnNmzfrqaee0tGjR5O8kLfSuHHjdP36a22p7bM9dMkjzjReooUaO
SdZ5dtz2/eyFqY4yioqJ09uxZTZgwwwcJkievZs6dz99rBgwerTp06mjNnjnx9fTVT9+nRrwyWhXx79+unDhgmrUqKF
cuXIpMjJSTzxhNWxkvTKK69oypQpHjUqdMWKFfrxxvHSOkT8d73NK8l7HKS6ADmE8LYZihlu66lmZ5e7Nh
VLVtW3377rdKnT291PEn/9+aBw9GOFCEChcbhlPmTmS8VVKH7NXVOMPOPuMp8uiOW9qwi8HmTnLkjyD
FxMRI+mdEUsOGDfX0008rU6ZHOMpXydnbuyQ6aOF9DbYKZnW2FpgnwIQLgoQzFq4aZFLlaC9xyOqM9fYm
BJGrS4ZYNPe9bCC9xqHGbWzE0cjWGY8SXRaXlBqt/zUlDcMrlBPORbmIbmU4sxBJX5Zm9Z3nd28qsYPX4CJ2Dlt20x
cNHWh4gKNGbgQrzBCWX5xeJEG+rnbOSJvcfHUx+GzKp07Nf09Qgi5R+JtPeE/jgeiXC5Z+bUkGzNPVadOK5KBHvRmx
+dMVVfI86D4O9mmlAmTzrxuUX24gLu4Oa2tyfjYyWSZOE2Ofu7I9H4RzDJTxpVBBJytLfBRbcV2uKzTKOq4omkLrf
9Tx73wu4aSz+OuY3xA1Uu8aQgaoENT9kyasy/5JEIfSfmOrfkwWqVDMymTUqjbzPWfZ4Og6TaSgI+i9j8gaxe3x
wZpj2GHoxb59vwbsRSwmfNyoLMpm5Gq6qOsjSsIatCcQCP8/zyEmD8dpukDgxvYp2oswFb7NnWe3zmsBsG3BMZC
jrz8BMmgDpTvhnz9tWWTvczn+rsjOTp4ZJpMajd5Pqe+uz/8M8iJumgcAdzNkjz2N+P1
/oGJCrKjyqzi2eXVL4OUxcH6xGeEL3/3iHb/LdXsB0vWEL43PGTNl+SRdpPyfHcQ7GSOSkgNAKCQ
u+R+t+7SnOOKdfu7+rbyuBTnDp3OP7xRLmUfUiCxcwsR8YJwuyyTvEKKsesVHVAWvDJj2B+
5J0fXpS4I1s2cLq8VqxRuQLRl8dOLSBlCwwfGfd6TQ94dlRw7hf7d3tcb8kydVU3W/cUyq3Yz+CvNrs3aUjG5AcsBSA6Yx/4
dIA7mwQxRLpKUVtPUzwaULDxeP+oHoSqgfNOd/fyd5vSmxmnVmmOxrwMKWsJPnLsSuL0KN5
nW+Xm/XjPzP37M8YXGK2bGe9eNWNOFgx/Cyi85DhIwjyRmpmvUiQZbUvpk3c7tjz2Mz7t6gmJuzEa3GZxnVWE3XHIjHyTyy2Y5
o0xX+gvGWXnE7c1K6O6SrN5L6NJ6dHeKllHSJvZ4cjSUFV/EWxRBmCGTHQlrU0yTU8SpWnbFWU6rTKmDqXBCVQqSfalcqQUGTY

HBzvf6LXLTs13O3funHOq+8mTJ/Xf//5X165dU6NGjVwWV8e/R1hYmL7//nsVLVpUTz31lPr376+WLVvqp59+Ut2
6dXXp0iWrIyYqQ4YMunr1qm7duiV/f/8E05rt9jrKE73wwgsqXbq0+vfv73L8o48+0pYtW7RgwQKLkiXXu7lkFsbG
x2rlzp/bu3as2bdpozJgxFiV7fAQFBWnbtm0JRpYfPnxuXYpUuXtuTxgl1BH4L169dr6tSpKl26tJ588km9/PLPLat6
8udWxkq1Pnz5q3769hg8fnujuXXZTsmRJHThwIEGxduDAAY9Y3Y3+OqIkSNatGiRy6ijVKl
SqXfv3po5c6aFyRJnx+Lsfk6fPq3Y2NgEx+Pi4vTXX39ZkCh5hg8friz+++/qVOnTnrttddUqVI
ltWjRwrmTrNXy5r+fPn6+2333Y5/tXX9lqR7o71x+KN3jwYLVs2VVIvvfSSnnnmGec5dpnaHC9DhgzKmDGjXn7
5Zf3nP/9R5cqv0/fffqyZcv0/dU7TRs2DA1btzYdsXanjnz3Mo7laXSbb7vvXFn12Jt7969KlmypCTp0KFDLvfmZTR5
0zxu9165dU7p06TRs2DA1btzYdsXanj171LBhQ508qVL58uXTvHnzXLduXcXcXExMjLy0ujRo3SwoULbb9eqt2tWrV
Ko0aNci7l8eSTT6pnz562HLX69GpnzAc562HLXWvn17jRkzRk8//bQiIiJUJOTHGjVqQWB0zSaNHj7Y6wmm
vMjIy0TeP6tatq48//48Hm//+/1rIH/yjk7e+UCFzebfL69PHiFs6gH2tPR06
UyfSPJM8xf3bNgmeCqNPaEPP6FiLnUv1v+M3FpxNqtNWVV8e/+OoFSunxLVTZLR8/h2MQ+G/MLPqbNyOMI3AX7//y1LQ
m/XrlClSRvJHW7fAyIzM0Zz8oXhR6sKd3UhpwfoG+RlGnI+bz+0TB8/IhFhVh8uMOqVXvLpJjWyf/L0ThMVXLhFSNF6IW
GTyQ0MLpOf6UGILiEeTBZ+DRlG8dcnsBVK4xrUMkT2RhKSyqg9Rq7XU7EcqQgMMuWCxjO9NFZVtAYDGzdW1mSZDUU8+7
rvJtYkE9CpVpm+oJhjbhcjtdnEYeNw9D7mNHmZTrtDzgt89UDLwyZ1QzMxE/gPXfcfbGmyhHXw/xSgGgBAGHISg//9HHM
fiMSKb2YX8vsDBfmMYwy9lamg/sQnt1LLmyHVuPVOMk1P/s/qQY9o5X7SJr0DqqJGFFMp0nSTUpLAw6lbLewOhWIRi8jbx
HRyoRv9OkENw+3/MbRZkaF8XABTkmelNHm6VyKcHZPydDlBsWUg1mZmK+HWB2AJxvKh0c7hKOSgn2j+zfVPfPl8f1Dg+
QQDaeT9Sqm4Ol8dmX8+3nYkBJsTM6+OGpxtS4sX4nNYXlyTVsIWIrwIXDrWNEmKfNNV1ogSxyGUmS9lDo0i+6hiEjjiSYNxL
uNQUIAQ5rgT68tuYOcKsLT7t3Nlo/iu//HNfx5hsT7PBi+PUqt97NL0tb3Gt+8Pwb2JnWTJt8vcEz/RLdd0Uc6XwLLZBB
DRnYf7F58pb5rUqQBXpU2Ez7hIILe9XusnB4E0DJn2a9v9kjCe89PVwn1X4HJJ/uX+pJ7ZYY2ADcZRfsT+LGg4WyL8fcb8

zZs1CREQEsrOzK6xL7YG/HL2qj53U/o7bt2+PpUuXomfPnqKjaCwDAwMkJibC2tq6zM7yGzduwNbWFo8ePar2TCy
s1WBmZmaIjIxUT2L617/+hdjYWMTFxQEAfvzxRyxYsEByjb3ldsMJPDvasHv3boSFheH333+Hl5cXAgIC4ObmJtl
dSk2aNMH48eMxZ86c1z7aIzU5OTkwMjKS7N+xubk5jh07JvlhBeVNmjRJvdPOzMyswt9vaGiooGRllW8ee+DAAbR
u3bpC34Xdu3dXZ6zXIrejzZX9jHj++0JqF71yVPp3XP7fmxSnYHt7e8PPV1RXTpk2rdH3FihWIiYnBnj17qjnZ60l
NTUX37t3Rrl07REdHw9vbbG3/88QdycnIQHx8Pa2tr0RHV6tati99//109SKZUYmIiOnfuLItdjlSzyXEq6PMmTpy
ImJgYBAUFYdSoUVi9ejX+/PNPPrFu3DkuXLoWvr6/oiLL3yy+/lHn99OlTJCUlYfPmzVi4cCECAgIEavc4cOHMWv
WLCxatAjt2rVTH3UvJaUHYXLVqFEjHDx4EG3bti1TWDt8+DACAgJw8BNas/Eo6A12F9//YVGjRqpX8fGxsLDw0P
9un379kK+KV9FpVKhXr16AJ4V2W7fvg1bW/ADOg2bBiGDRuGGzduYNOmTfjss8/w0Ol
mqv88UlJQUIChQ4fKtqgGQPI7AwMDA7F8+XKsWrVKssW/yoSHhyMiIkLyO+3K92sZNmyYoCRvTm5Hm//
veufNm4fFixcLSqVZYmJiJiXrgmtQdKycnJ+Prrr1+47ubmJskjzaXs7e2KC//znP9DS0lLvhp44cSLMzMxEyu
jadOmFY6eAc8Gyoho3kzScOvWLezdu7fSXxs7e2RkpKC//4w7w9/dHly5dKytAxWkqzzSNm2vprAfB79+7w9
FtXegf//+Fd77+OOP8OdyN23jlS+zcuVOnBcMRzbHkJjPkKFeRGGX0KWTV3Yzdi+hxQ
wrBF137590aVLF/XRZqmprOl07969UadOHbxvqs8aYq
6/aCtrY179+5VY6I3Z2pqirs1JdbAtj4MxefJkrFVRkpuzm8JaFNjRVdgbH2VlPRWprN8eURBcYL73v3N26Lbs6a9LAmzabMsqAzoRAmTecYWOHbFVWfJwyfApFlAEqlR4Zv1Z
1aoXr16+jpKQETk5OouNpJycHPVUW319ffX00o8++ggTJkjGUDdNWA0kTm0QPGRQCEerBrIEagrQM
hCgsL0a1bN2RmZuLDDz8U9kCXhbUazMPDA7Nnz3+GTo6OmX6naSkpEJPrWJkWEPBFg5YB6uRrqF375
YtWoVPDw8JLsj7KuvLfvnix8ODBSvv5SO8Ttz3PNuWbbQbkGuNOhGboAEDBoiAlfECTPAr03QuMKbuPV3SOO7sugnuPStTk
fW1lZITU1VfJHHmytZ4Z3EchUdHY2jR4+ri1f5KHfrLlUYy6YKHfUpIG9eRTSbiFOQuTwOVBR
6OlZhkjfj5+eHgoJCdQ3X5+xRHpQB1ZQ4iZiL/0TyGl+lHHrp4pKmEwaKYVYtSmgHD2uXdp8Kozp4FJ0QxbSgyhI
OptC7YWVlhevXr8PCwgL29vblV/a8FkzQNl5qVKvT/KpCjNqE2xSSSnDn0fYH+/AOtWmNr0BvYjJyDL82bcXCtTlDjd
OTk7o1auXsEzssVaD3bt1DU0KzdsPDRkoirOauQnVumsXdq1MxYJLGKAGJ7v/Q8vvbwAAhsGpPs7QKjMicz0ThPmvtCKM
LFi3CggULYGtri0aNUXiC1BvUGBgY4cOAAnfuLDqKxhswYAPKYOGSFwsQv359We20o6pTvihRuRLKCOfuYOnSpXj
69Cni4+MFJdMMt27wdqZNmxwAWFob8/HwMHjwYa9euRauzt7HTq2Dy5Mk4Gvffee6/MwMFhIVxcxOD
q6ooVK1YISlhRaXPz2rVrwf8GenAuNw8HOhn6enaH7rGtUrhtbjcmFZcuHhbtzNmXCyubY
9uuiIGiU0NBRaWlqYMmUKYmJi4OXlBZVKhaKiIxbtgxTp04VHVH2yt/jlZSU4h00HMGDB9DR0cG2bdvg7e0tMF1FHOh
UM7GwRsjLy09evWgpaVV5v2cnBzRUq1cPtWvXFpRMMyMiyiVSJRt2hRt27Z9aeFUn
w8HBJ7STQVGPGjHnpFtOnT0fLyjwg+li5d+sLqkbyvt8+Nz+shYGOzs7AQEv/HPiBE2d0Iw
WZAtrWVlZcHJygamZNmgRbVbQbN26KOv05KSsL06dEYhWzy8gmCAEyd
O4JtvvkFwcDHwEpSR6NVNTU1RHR8eC1D0RHh8PP8PHr2tDHqVY0VrRhBe3cdTLyE
tOo5GKP8AQalUwsTEBC1btsSCBQsQCYmKFnlONCpeikKCO3fuYONpqpQIhGmpqY
4fvw4mjdvLjKbKrF27hZdu3YhYYwmjdvLjrKaKi0gbDUuwLR9Vj2LBhiI2Nxc
A/7voLb9bid6ctrY2pkyZggOOHoUJQJOHToUJLjc+7u7lizzg2aNWsmo0OX
AgQOYN2+eZHsGFhYWv1zDNZ4NIyMjD16lmgNsS2H2+ujdMw+NIyMjHD16Vk6ejo9bhHWo
1spvwrgmSk5Ph5OQkmdfgxL5emdflBzr17NlTUWTUDLNsWTJEnzxxRcqeE6ZmZnRufuJmnTt27ZMW
sXLlSUkd1XsbZ2RmPHj0uRUVdHR0KhxPlGJvmRUrVuBf//oX9tZlnfWuVCpLLrT0PpUV+++u2PvhUXZQ8FF0vJcqKir
CsWPHcOXKFQwfPhx6enzffs29PX1JTnlVm5+/fVX1FSaXC7ar9fXPX1JTnlJ8sSQoqQ30n
2Nxw/fhxhYWFwdnaGnZ0dRo4cihSFDhoiO9UoWFhY4cOAA/vrrL1y+fYqmbkT/P0tI
SqampAhK9WH5+PmbNmoWWKNm5qlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVvWXLlqlq3pX355Zd4d4+PAhdu7cCRsbG3dWJfaDSdVv

yEk5OTggICMC0adOwa9cunD59+pUXEyIVFxdX+nTw1q1bPJL2jshl6mBeXh60tf/vUuLevXtl1ouLi8v0XKN3S0t
LCz4+PrL5fpGL/Px86Orq4pNPPhEd5ZWsrKxw/fp1WFhYwN7eHhEREXBxcUFkZCQMDQ1FxyMBFi9ejOnTp2PRokV
o165dhWKwFK+H5CwmJkZ0BI1VWa+y8uujRo2qpjSv79q1a2Vec6DTuzd58mTExMTA1dUVxsbGkihi8ygoycLChQt
f+7MLFiyowiSazcnJCVFRUTAyMqpQzCxPagXM0l0y5TOXlJRAoVBIbps48KzoUFxcrC5MREREIC4uDjY2Nhg/fny
FHWFSMWTIEBgYGOC7776Dnp4eUlJSYGJigv79+6Np06ay2+0oNUVFRVi8eDH8/f3RpEkT0XFeqnnz5li6dCkGDRp
U6XpERATmzp2Ly5cvV3MyordXr149DB48GP7+/vjoo49Ex3mp0NBQaGlpYcqUKYiJiYGXlxdUKhWKioqwbNkyyU0
Qpqr3/K7h56+JpHw9RKQJONCp+ujp6SE8PBxeXl6io6ixsEZEagsXLsSMGTOgo6PzymKm1AqYsbGxL1xLSkrC559
/Xn1hNNzt27fh6uoKLS0tXLp0Cc7Ozrh06RKMjY1x/PhxNGzYUHRE2dPT08O5c+fQrFkz0VFeaurUqTh69CjOnDl
T4UKxsLAQzs7O6NWrF5YvXy4oIdGbi4yMxKZNm7Bv3z5YWFjA398fo0aNgrm5uehor5SRkYHTp0/D2tpaVj1H6d1
52fUQAHTr1q2akmiugQMHYtOmTdDX13/lCYN69eqhZcuWGD9+/Ct3YJG8caBT9bGwsMChQ4dgZ2cnOooaC2skO1Z
WVjh16hSMjY3LvJ+bmwsnJydcvXpVUDKSkry8PGzfvh3r169HcnKyZJ/Q5ubmIiEhAXfv3kVxcXGZNSluby9VWFi
IHTt2IDExEcXFxXBycoKvr6/64oH+ntLjfVIaI16ZrKwstGnTBrVr18akSZPwwQcfQID09HatWrUJRURGSkpL
QqFEj0VGJ3lh2dja2bNmCTZs2ITU1Fe7u7vD394e3t3eZI9BEVLOMGTMGK1asgJ6e3iuHfT1+/BgnTpyAg4MDh8x
oOA50qj4bN27EwYMHsXHjRsn0XGdhjWRHqVQiMzOzwq6YrKwsNGnSBE+ePBGUTDOdOXMGaWlpUCgUsLe3R9u2bUV
Heqno6GiEhYVh9+7dsLCwwKBBgzBo0CBJ5o6MjISvry/y8/Ohp6dX5siGQqFQN6GWmuzsbHVhOyMjA+vXr0dhYSG
8vb3RpUsXwek0w7p16/Dll1/C19e30h453t7egpJVdO3aNUyYMAFHjhxRN+pVKBTo3bs31qxZo76gJJKzlStXYsa
MGXjy5AkaNGiA8ePHY/bs2ZK4oI+KikJUVFSlD2jYKLtmys3NxYYNG8pcv/n7+3PHlCCpqalo37498vPzRUehKmR
qaoqoqCi0bNkSAGBiYoJTp06pTx9cvHgR7du3R15ensCUmqFt27a4cuUKSkpKJNNznYU1ko3Spzw+Pj7YvHlzmYs
DlUqFqKgoHDlyBBcuXBAVUaPcvXsXQ4cOxbFjx2BoaIiSkhLk5eXB1dUV4eHhMDExER1R7datW9i0aRPCwsKQn5+
PwYMHY+3atUhOToa9vb3oeC/0wQcfwNPTE0uWLJHEzdmrnDt3Dv369cPNmzfYsHlzhIeHw8/PDA/n5+VAqlcjPz8e
uXbvYSP0deL5HTnlS7ZGTk5Oj7qVmY2OD+vXrC05E9PdkZmZiy5Yt2LhxIzIyMjBgwAAEBATg9u3bWLp0KczMzHD
48GGhGRcuXiigoCA4OzvDzMysQp/RPXv2CEpGopw+fRru7u6oW7cuXFxcUFJSgtOnT6OwsBCHDx9WT7Ck6qNSqXD
+/Hkez9ZwdevWxdmzZ2Fra1vpenp6Otq0aYNHjx5VczLNI8WWRSyskWw835y+/LdtrVq10KxZM3zzzTfqhpH09ww
ZMgRXrlzB1q1b0aJFCwDPnriNHj0aNjY22LFjh+CEz3h6eiIuLg59+/aFr68vPDw8oKWlhVq1akm+sKarq4tz587
JZkdPnz59oK2tjVmzZmHbtm3tf28NzcsH79egDPJvScOXMGJ0+eFJyUiOjt7d69Gxs3bsShQ4dgb2+PsWPHIEW
IEWWmbP7xxx9o27at8F3yZmZmCA4OxsiRI4XmIOno0qULbGxs8P3336wuPLBcVFWHs2LG4evUqfvvtN8EJNc+pU6f
w448/IiMjo8LPhN27dwtKRdWNA51qNhbWSHYsLS1x6tQpNGjQQHQUjWZgYICjR4+iffv2zZd5PSEiAm5sbcnNzxQQ
rR1tbG1OmTMGECRPQvHlz9ftyKKwNHDgQQ4cOxeDBg0+jr6yMhIQHOzs4AnJ2J69i
xo2S+N+TI09MTO3bsUO/IXbx4MSZOnKi+oc/OzkaXLl2QmpoqMCWRZjMwMMDQoUMxduzYCr8DSxUWFiI4OFj4IB9
jY2MkJCTA2tpaaA6Sjrp16yIpKalCU+/U1FQ4OzujoKBAUDLNFB4ejlGjRsHNzQ1HjhyBm5sbLl2iix07dsDc3BwA
T0msQDnSqfk+ePKm0DULTpk2rPQs7r5LsXLt2TXSEGqG4uLjCeXXgWcGq/A8vkY4fP46wsDA4OzDzs4OI0eOxJA
hQ0THeqHnG9d6eXlhxowZSE1NhYODQ4W/byn10QKeHfUzNTUF8GzKla6ubpnjfkZGRnjw4IGoeBrh0KFDZUazf/3
11xg2bJi6sFZUVMTj7kRV5P79+wCePSQo7WtY+t7z9PX1UbduXeFFNQAYO3YsfvjB8ybN090FJIIfX19ZGRkVCi
s3bx5E3p6eoJSaa4lS5YgNDQUEydOhJ6eHpYvx5aT8cJbyE6X4d0x
UdZRKZYUeZc8TecH+vH/+85/qr4uLi7F582Y4OjrC0dGxwgOaZcuWVXc8EmzKlCnYs2cPQkJC0KfQ0KlTJyCsTFxWH
GjBkYNGgQvv32W9ERNYquri7++OBlpaGHj164M6dO6IjUjXiQKfq0blzZ2hra2P27Nm9V9hcV0c+QO9ZIdl7VqJfejVWrVqF///5o1qwZmjRpAoVCgYyMDDg4OGDbtm2i41Wgo6MDf39/+Pv+Pv748KFC9iwYQO
WLl2K2bNno6fv3pIZcS6l3X5vw8/PD3Xq1AEAPHr0COPHj1fv4Hh+pxURkdzExMSovy4pKYGnpyfWHCBCxdulR0PI1Tv3599W79xo0b4/z583B
VUVJSUpnXbdq0AQCcP39eQBqSmpCQECgUoKSlB7dq1MWHCBCxdulR0PI1Tv3599W79xo0b4/z583B
wcEBubi6P3dZAlpaWOHjwIAc6VbGzZ8/izJkzFXbmisQdayQ7bNRbvY4cOYL09HSUlJTA3t4evr6+GRiIsLEwyhTUAiI6OxqRJk3Dy5Eno6+wXWcvLy0OnTp2nRSDvY4cOYL09HSUlJTA3t4evr6+Eh3ptalUKkR
GRiIsLEwyhTUAiI6OxqRJk3Dy5Eno6+uXWcvLy0OnRSVDCyo0ZM+a1Psd+Im9PS0sLmZmZ3BVIJAH
ld4wSyUlBQQGuXLmCkpIS2NjYyGL6uBwNHz4czs7O+Oc//4nFixdj+fLl6N+/P44cOcLP44cOQInJycOLyCqAu3bt0uQv
++ugj0VHUWFgj2WGj3qol16KPnHh7e8PV1RXTpk2rdH3FihWIiYmR3HFbqnpKpRJ9+vRR7wqMjIxEjx49yuwKPHj
wIAtrRNVADoW1qKgo9OzZs9K1VatWYdKdKkSdWciEQZOHDgKz+jra0NU1NT9O7dG/369auzsLDAwMHBwcEBGzoaIgjR45UmF767bffYv78uZUOXyDN9Dq/P4qLi3H
37l3ExsZi+vTpCAoKQkpKiiH9PT4ebmhoyMjGpORqJxVyCRdMihsGZoaIgjR45UmF767bffYv78+ZUOXyDN9Dq/P4qLi3H
peo0bN8a5c+dgY2NjebmhoyMjEf2djbc3NwQFxeHnT3YsmSJfj111/RMn0fF
Iojp37gxnZ2fRMTRChw4dkJSUBAsLC9FRiDRat27dXriWl5eH7du3Y7Xv4etDiyskezMmjUL9erVY6PeKsKiT9W
Fn6rn6emJ+fPno0+fPno0+PnjvvffKrBUWFmLBggXoXo27evoHRERDWTgYFBmdcjRowQlOTNTJ8+HdnZ2XB2B2doZKpcLhw4f

RoUMH0bFIwgwNDSVVIJazzz77DIGBgbh16xbatWtXoRDv6OgoKBmR5ouOjkZYWBh2794NCwsLDBo0CBs2bBCShT3
WiKiMyZMn49ixYzh16lSlRR8XFxe4urpixYoVghLKX1ZWFpycnKClpYVJkybB1tYWCoUCaWlpWL16NVQqFRITE9G
oUSPRUYmISGJe9Ps3JCQEXbt2hYuLi/q9KVOmVFcsohpJqVS+cE1UryciTXbr1i1s2rQJYWFhyM/Px+DBg7F27Vo
kJyerWyKIwMIaydKpU6fw448/IiMjA0+ePCmzxidwfw+LPtXjxo0bmDBhAg4dOoTSH8MKhQLu7u5Ys2YNmjVrJjY
gERFJkqWl5Wt9TqFQ4OrVq1Wchqhmu3HjxkvXeUSU6N3x9PREXFwc+vbtC19fX3h4eEBLSwatViVYY3oTYWHh2P
UqFFwc3PDkSNH4ObmhkuXLiEzMxMDBgxg8+93gEWf6vPXX3/h8uXLKCkpQfPmzTmWnTYiIiEgmsrOzYWxsDAC4efM
mvv/+exQWFsLb2xtdunQRnI5Is2hra2PKlCmYMGECmjdvrn6fhTWYlPP/0UEydOhJ6eHpKTk2FpaYlPP/0
UZmZmWLhwoeiIGONFHyIiIiKiss6dO4d+/frh5s2baN68OcLDw+Hh4YH8/HwolUrk5+dj5d165d8PHxER2VSGOcOHE
CYWFhiIiIgJ2dHUaOHIkhQ4bA3NychTWiN6Wrq4s//vgDzZo1Q4MGDRATEwMHBwekpaWhR48euHPnjuiIRERERNX
i1q1b2Lt3b6XtMZYFr1ixs27YN+/btg5ubG9avXw/gWc/iM2fOTEmmegoIChIe
HIywsDAkJCVCpFi2bBn8/f2hp6cnJBMLayQ7TZo0wbHDB+Dg4IDWrVtj9uzZZGDZsRwDRxQUFGD69Oq9hihwScC3Lx5EwqFAu+//z4AICEhADw6en
U5aKiouDt7Q1LS0tcuHAB8rVq1wvXr11FSUgIAPjpQEA6en
p6NixI3Jzc8UGJdJwFy5x5cwIYNG7B161kb5uaid+/e2LJli8Qdqe3tbb7XnePEYEyKJ6tKlC44cOQQIAGDx4MKZx48Zh2LB
h6Nmzp+B0RERERNVjzpw5cSCAwMxPnz5+/Hee+/+/hp59+ws2bN9GtWzf84x/Ev8ghH8FBHhSMRjbsbeHCUXmG
66nUjIyM8ePBAVDyiGsPW1hbBwcG4desWwN+6SwHd6yR7OTk5ODdHcXEkQkJCEBcxBsbG8ybN499wIi
IiKhG0NPTw9mzZ2FtbQ0j0jIyPExcWhZcuWSE5ORv/+XH9+nXREYkYk0klKpRFtvlZUFc3N
zqFQkTGJqJpoiw5A9KaefxqkVCoxZMjw5U2AiIiIouqnuqn6uLx48fAwDMzqZ1Uo1tWbw3//V2Q0Io3
n5+eHOnXXqAAAePXqE1ePHQ1dXFwFdXFwDU/y6JqGZGYY1kqbi4GJcvX8aIg4CJWEa8KwsLWLAGfgTVqVHXFISLBeBSUZOfkyZMYPnw4bty4gfLvq
29vbw8vJCYGAgzp07h0ePHomORkRERETVwsLCAo0aNcKvv/6KhkZG7du3h6qqKq0aP7aicmd6uHDr0vmnVtgycfGJldlsSy9DzG8AsTjr2xVT40ZJ09
FgluuiYiIqEa4evUqHj58EDHRxQUFGD69Onq9hihwScALAQHZGIiEjjsbBGstOmTRt88MEHWLhwIczMzBQKMq
sGxgYCEpGREREREREREREREDUJC2skO7q6ukhOTojNY3oKEREREETC3Lx5EwqFAu+//z4AICEhAT/88APs7e3xySefCE5
HRERUMyhFByB6Ux06dsMDly5ZTTHGhZWcNmmTQDOAZEZGQFKp66jOKKmo4RV0nTnUpavDUkYBRRWcxZ8VD9UKHxBiFpVXTgjhHRnGdDZEM2TXRT3toHCD+LC7oWqb08u2VT3MuJbAVRVrmNrNNjCf9XTVQTDRmMRFYj2dCNdvwARmIw9rRJnTtvdH7DOsxhrUuogfthT3WFEXEL4ZkOvOcmWnkA1pFJbL6EGY2NjjkQGH2N8QwcTCUVl1HImZBCB4mPc4d3e7rDE6+fe3xq66GVEKPTjvjy5Cw/DvYChSCfcTfAE5J
HRERUMyhFByB6Ux06dsMDly5dFxyAiUISavjw4YiJdYaHiQuvBKx4iWHBXDtwPzz8Lfv0sYifFNivLFvEeKjHQGZEDJBR4UcivUkTvaDYKOyYk6fdkLeF6Dvd2LJO20YjnKdHJrdvmFLX
7Ij09HcbGxti5cyd69OMzpsaaWhPRzbmAAgFCqdPAwqCLRj01IvVH9MKOzTT7nzBfbTm3iDzl9mvWsnsLiIxxfdGyeMYdwohzcVhj5AMevZx0QGERKHxnGQn7Mk4xKKRMgnwmi9hw3cCyGZFPsEjmR
HRkZGL23gS0RERKSprly5go0bNN2LLli24c+cOevfujYe6hoHp5Ff1RUWCPZ2bx5Mz7++GPo6uqKjkJEREQkGQ8
fPsT27dsxd+5c5c5O5bmQVSiY5EREREskKj8d6jWRn+vTpaiwIYYYp9+/ahqKhIdCQiYYjYzF69GiQyqUUZfxbB2toarq6
ZGDhwIOLj40XHIi7IxIqhFYWCPZuXPnDnbu3Amj5Akt0+MHToUZ7FjKEReRULW7evIlFixbX8+/GkN22P4ULjsOzyZTq0eP7+9M2xt/+KUrpqw8JdA9rJEyMD3MgKy/9vtNGrj9CXzY+FeO39jeJcO2sbpkiiMjqMMlEG7FMQXzpfJAXxZMOOdm8+KD6pflN7wm0Xm66x16ebM+a4voi+qVyqQRqe8VanaYAHUw8WPtQGQlbWIkJsqQNl3TjwDV39BqKgNSCnnEtBVKZmAsRaBtrqGrlKhVtpNb6kLkB0yNHmYEdm4Ah2IoN9BPTqkSlCM08lAUBb93HJ3gN4Mb6S3ZF+eyO

ERESyw8IaERERkYbLzMzE5MmTYWVlhTp16qBJkybo168foqKiAAB37txBnz59AADXr1+HQqHA2bNnBSYmIiIikgd
t0QGIiIiIqOpcv34dnTt3hqGhIYKDg+Ho6IinT5/i0KFDmDhxItLT02Fqaio6JhEREZEsKUpKSkpEhyAiIiKiquH
p6YmUlBRcuHABurq6ZdZyc3NhaGgIhUKBPXv2wMfHBwqFosxnunXrhqCgIPTs2RM3b94sU4QLDAzEqVOn8Ntvv1X
Ln4WIiIhIangUlIiIiEhD5eTk4ODBg5g4cWKFohoAGBoaVngvISEBAHD06FHcuXMHu3fvRteuXWFlZYWtW7eqP1d
UVIRt27ZhzJgxVZafiIiISOpYWCMiIiLSUJcvX0ZJSQns7Oxe+39jYmICADA2NoapqSnq168PAAgICMDGjRvVn9u
/fz8KCgowePDgdxuaiIiISEZYWCMiIiLSUKUdP8of73bfn5+uHz5Mk6ePAkACAsLw+DBgyvdCUdERERUU7CwRkR
ERKShmjdvDoVCgbS0tL/932rYsCH69euHjRs34w4AD8/f3fQUoiIiIi+WJhjYiIiEhD1a9fH+7u7u7li9ejX
y8/MrrOfm5lZ4r3bt2gAAlUpVYW3s2LEIDw/HunxXrYG1tjc6dO7/zzEREREREEREREUkdqBlXYcmEkZRxkUNMrAJNvzB9xa5r+YSnu6+XdueXoEPOzkkhEAxhwAdBbk/Sv6NljJRJSfDXZnrIzJOi3SXFilTZmSsdr7lLpgDgcuZ+9o4w2VZUeykPR2j9b9//H+X2+W2fSbDOM3P/iKXCcOVAsQwB8Pic3tV5A0fSwCMIXi8+XHzHmeMEHdo80COAbUTFf4QO5tq8b5A+IIaTkChiTS2fsT9UY7O0ysKcwQc
ME5xRa5qQ1XqE6BN+cGHJHIVxMPm20FxjF+H3PfczhrANE8zWw1YVTu7PB8qa8H6ltuMCWn/KrCfw/9Z+NSgRB1yzeUZ+QXM//9S0sweX9zUl0PG/+PpE4hENQhjYpGeUXuLPHC6ve3yXf3Cva3eXh+PvvMTd8vpMfHNX3Svk9upSyDmwO1jOzcUBx9mk6eh
oLbCwRkRERERERERE9Bb+HxvrWxlJMRn1AAAAAElFTkSuQmCC",
"text/plain": [
"<Figure size 1500x500 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plt.figure(figsize=(15,5))\n",
"sns.barplot(x='City', y='AQI', data=city_median_AQI).set(title='City vs Median AQI')\n",
"plt.xticks(rotation=90)\n",
"plt.show()"
]
},
{
"cell_type": "code",
"execution_count": 21,
"metadata": {},
"outputs": [],
"source": [
"import datetime as dt\n",
"df['Date'] = pd.to_datetime(df['Date'])\n",
"df['year'] = df['Date'].dt.year\n",
"df['month'] =df['Date'].dt.month\n",
"df.drop('Date',axis=1,inplace=True)"
]
},
{
"cell_type": "code",
"execution_count": 22,
"metadata": {},
"outputs": [],
"source": [
"city_median_AQI_per_year = df[['City','AQI','year']].groupby(['City','year']).median().sort_values(['AQI']).reset_index()"
]
},

```
{
"cell_type": "code",
"execution_count": 23,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>year</th>\n",
"      <th>AQI</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Aizawl</td>\n",
"      <td>2020</td>\n",
"      <td>23.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Shillong</td>\n",
"      <td>2019</td>\n",
"      <td>44.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Shillong</td>\n",
"      <td>2020</td>\n",
"      <td>49.883721</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Amaravati</td>\n",
"      <td>2020</td>\n",
```

```
"      <td>54.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Talcher</td>\n",
"      <td>2017</td>\n",
"      <td>60.578431</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>98</th>\n",
"      <td>Delhi</td>\n",
"      <td>2015</td>\n",
"      <td>303.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>99</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2016</td>\n",
"      <td>306.098712</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>100</th>\n",
"      <td>Gurugram</td>\n",
"      <td>2017</td>\n",
"      <td>314.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>101</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2019</td>\n",
"      <td>455.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>102</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2018</td>\n",
"      <td>509.000000</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>103 rows × 3 columns</p>\n",
"</div>"
],
"text/plain": [
"          City  year         AQI\n",
"0       Aizawl  2020   23.000000\n",
"1     Shillong  2019   44.000000\n",
"2     Shillong  2020   49.883721\n",
```

```
"3    Amaravati  2020   54.000000\n",
"4      Talcher  2017   60.578431\n",
"..         ...   ...         ...\n",
"98       Delhi  2015  303.000000\n",
"99   Ahmedabad  2016  306.098712\n",
"100   Gurugram  2017  314.000000\n",
"101  Ahmedabad  2019  455.000000\n",
"102  Ahmedabad  2018  509.000000\n",
"\n",
"[103 rows x 3 columns]"
]
},
"execution_count": 23,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"city_median_AQI_per_year"
]
},
{
"cell_type": "code",
"execution_count": 24,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABTcAAAOuCAYAAADSDSMSAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjYuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8o6BhiAAAACXBIWXMAAA9hAAAPYQGoP6dpAADZIUlEQVR
4nOzdeZiVdcE//vfIjg4gKjOSophkKpaGSoIBLpB7RuZTtKipmSxKrilPCpWo+IgUuGQqqIi2qKktJKaOj5KJmIn
6aFnkkkxYIqLseH5/+ON8HRkUBGbmltfrus51eT7355zzvo/DLO/zue+7olQqlCXAAAAAAUDCNhAAAAAIC
JAAAAAKCTlJgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
KFSuybNmyxo5ReC1atEizZs3WaK5yEwAAAADWQalUSm1tbV577766K5bXGjvKh0aFDh1RXV6eiouI95yk3AQAAAGAdrCw
2O3XqlLZt275vIcfqlUqlLFy4MHPnzk2SbL311u85X7kJAAAAAB/QihUrysXmFlts0dhxPhTatGmTJJk7d246der
0noeou6AQAAAAAHxAK8+x2bZt20ZO8uGy8v18v3OYKjcBAAAAYB05FH39WtP3U7kJAAAAABSSchMAAAAAABSQlUSm1tbV577bXGjvKh0aFDh1RXV6eiouI95yk3AQAAAGAdrCw
AAACADaDHmTc02GvNvOTrazX/wgsvzG233ZZnnnkmbdq0Sa9evXLxxRdn+fHlGjBiRAQMG5OnnTJz5sz06dOnyPp48aNy4cOHDh0tbU
M5Zdfnl133bU85+qrr86UKVPy2GOPzcnJbU1ZtZt275vIcfqlUqlLFy4MHPnzk2SbL311u85X7kJAAAAAB/QihUrysXmFlts0dhxPhTatGmTJJk7d246der
kampqMmTIkDz88MOZNm1ali9fngEDBuTNN98szxkzZkzGjh2bCRMmZMGamurk7//v2zYMGC8pyFCxfmoIMOyrn
nnvuer/e9730vc+bMKd/++7//e73sh5WbAAAAALCRydOTp1ap37EydOTKdOnTJz5sz06dMnIp48aNy4gRIzJw4MA
kyfXXX5+qqqpMmTlJ510UpJk+PDhSZL777//qaeeyqqqqvTp0yfTp0/PPffck6eeeiqrqqrSp0+fTJ8+PQ8++GCe
eOKJrKqqSp8+fTJ9+vTcc889eeqpp7Jq1ar06dMn06dPz7333punn346q1atSp8+fTJ16tTcfffdeeKJJ7Jq1ar06
dMnU6dOzV133ZWnnnoqq1atSp8+fTJlypTcfvvtefrpp7Nq1ar06dMnkydPzq233pqnnnoqq1atSp8+fTJp0qTcfP
PNefrnl133bU85+qrr86UKVPy2GOPzcnJbU1ZtZt275vIcfqlUqlLFy4MHPnzk2SbL311u85X7kJAAAAAB/QihUr
XMFlts0dhxPhTatGmTJJk7d246der0noeou6AQAAAAAHxAK8+x2bZt20ZO8uGy8v18v3OYKjcBAAAAYB05FH39
WtP3U7kJAAAAABSSchMAAAAAABSQlUSm1tbV577bXGjvKh0aFDh1RXV6eiouI95yk3AQAAAGAdrCw
AAACADaDHmTc02GvNvOTrazX/wgsvzG233ZZnnnkmbdq0Sa9evXLxxRdn+fHlGjBiRAQMG5OnnTJz5sz06dOnyPp48aNy4cOHDh0tbU
M5Zdfnl133bU85+qrr86UKVPy2GOPzcnJbU1ZtZt275vIcfqlUqlLFy4MHPnzk2SbL311u85X7kJAAAAAB/QihUrysXmFlts0dhxPhTatGmTJJk7d246der
```

zZkzGjh2bSZMm5WMf+1h+8IMfpH///nn22WdTWVmZJFm4cGEOOuigHHTQQTnnnHPqfa1bb701J554YkaPHp39998
/pVIps2atXd7VqSiVSqX18kwAAAAAsJFZvHhxZs+ena5du6Z169Z1tjXlcvPdXnnllXTq1Ck1NTXp06dPSqVSOnf
unOHDh+fss89OkixZsiRVVVW5+OKLc9JJJ9V5/P3335/99/99tsv8+bNS4cOHcrjy5cvz/bbbb59Ro0bl+OOPPX+M87/W
+vpPD0gEAAABgIzd//vwkSceOHZMks2fPPTm1tbQYMGFCe06pVq/Tt2zfTp09f4//8T//jObbbLJJ9thjj2y
99dY5+OCDVzm8/YNSbgIAAADARqxUKUKuW0007Lvvvvum+7duydamtrkyRVVVV15lZVVZW3rYm///3vSZKRI0fmv//
7v/OrX/0qm2++efr27ZtXX311nbMrNwEAAABgIzZ06NA88cQUTufNGv2NSSAESMg+AGDQWdxYm9nrumpwMnuEvSTPzTNCMRSz9Ka871fsMHRmSBAK1OWAWjRmc//
f+EJ69OiRiRMnpqKiIj//+c/XLXiUmwAAAACw0Ro2bFjuvPPO3Hfffdlmm23K49twXV1UmyyirNuXPnrrKa871svfX
WSZJddtmlPNaqVavssMMOeeGFF9YlehLlJgAAAABsdEqlUoYOYOHZrbbbrst9957b7b27Vpne9euXV5Nd9XZ1p06aVx5Y
uXZqampr06tVrjV+nR48eadWqVZ599tny2LJly/KPf/wj22233TrvR/N1fgYAAAAAoFCGDBmSKVOm5I477lkoFPzVFRbKV
5hWb79u3Tpk2bVFRUZPjw4Rk9enS6eGnwa2tUbmueeZmvsNfZQYtO0Wnbtm0GDDRpUfp7a2trU1tbmueeeS5LMMmjUrlZWV6dKlSzp
27Jh27drlW9/6Vs4///222W677XLJJZckb74xS+u834oNwEAAABgI3PllVcmSfr161dn/Ic//GE++MGTcJXGe
ddVYWLVqUwYMHZ968eenZs2fuvvVVGTVqPl+nz59VmeSy65LMjJcmHzkJIzgv3xIBF7QnHV13MrNE3NyeHodGiuclvhS4iSVTRIXh8kmdBh97uTGzLgYV2aX1iNyvwq
RgQMHJkmuv/76VFVZcqUKTnppJOSJMOHD0/y9grN+rRs2+8uWLcudd96ZoUOHpqKiYp33w8pSpV1eSAtG2ai/8ZwHHEI
IzZ8/P0nSsWPHJMns2bNTW1ubAQMtUqffv2zfTp0z/w693xd5u9mHpDhdyqVb6+aIFlWLcudd9ZoUOHpqKiYp33w8pNAAAAANj
pp52WfffdN927d0+S1NbWJJkmqqqv+yCuvfbafPazMH1d8F9OeS3bbIOPbKS6r+ST2oV8iG3gqsxaqQVP1XLmmrfvq5Ljhwx
HV9n27kPHS6XSBz6c/KWXXsrvfve7/+5GvNAQgSUXL/s7FJpF+FfxA9ZG84fjp6fZWuVAyd/KZE5LfTvOW9YfBtnBh0A8L
OXWU155qaOHFitthiixxBEfPPC7KDcBAAAAYCNTKpUzb
3JjgXqNHAI9n6QUqkPXIp91fWrvZ0mWkPJMpUTBAf/3oaaTsGe9ytWKMtVoHBo6sXhLN7T0k/
HZ/To0enWrVu6deuW0aNHp23bnQ3bthk0nSVwI/m0zrO2/0Ih6wVKlukJYp/6qmsEXd9JoZgPKJXtqm6sXUmQZcqwlU/RY2Ue
5CQAAAAAUUvPGDgAAAAAAH0Y1ffo22Gv1faBmreZfeOGFue222/LMM8+kTZs26dWrVy6qd2CVvfjNoBw5n5TRNOOY5klJkJiMBW/tJFpjro
K1VdfnXnz5qVnz565/PLLs+uuu5bnXH3ZZnBhgwXMNiDsnpL8pRfuLmnNVZJrS3ahpHBtq/XSZ7mmKrOB9aGUa66HsWAqPRVVTMO+NNBY7Oz73XL8z/9
bbvnBD36Q/fbbb53ObG8OFcGbGQFH+qYPxLawzQVNVVfYoVk2ZVvoW+Sim6xg8DhxR8PaWGmVNkzqJhoOQbPIu2tGYe3ecxp/VacGMVppkbCQsGlWWvfpeanlQlPVzbXcjePOZxcqO2KOLxQsiKoHnC4DtOX+pCWhQGu+zbvBs8l
UV1enf//+WbBgQXnOwULc9BBB+Xcc89d7WsdeuihuG0oUEkY+cYgZjhOxUkRSUHPFjhnSuqk4ItdPDDNfCAVfjpDgvM/0MGKq5pqmv4DaIxXvwE9
AAABshBYvXpzZZs2ena9euad26dZ1tTXnl5ru98sor6dSpU2pqatKnT5T5+USqV07tw5w4cPz9ln50kWbJkSaqqqnL
xxRfnpJNOqvP4+++/v/9e9/Z6uttsoDDzyQzmzM0mSBQsWpF27drnw+rnxJsdwAEH1Jvnvd7Xd7JyEwA
AAAA2cvPnz0z0+SdOzYMUkye/bnz0+ae9euad26dZ1tTXnl5ru98sor6dSpU2pqatKnT5T5+USqV07tw5w4cPz9ln50kWbJkSaqqqnL
nqqoqPXr0WOfczrkJAAAAABuxUqmUqU0047Lfvuu2+qKKKKqqQMqnyvPPP7/Gz/Gz11prcVXSF+06blc5/7XCo
rK7PJJpukqqoqU6dOXeXcnB+ElZsAAAAAsBEbOnOnRfrnRonnjjiiiiLf//u3IiqrzlFV+1QqYx1Z7OSKGsnU1GdvrtloqrfjBf3y
3f/PII4/kc5/7XA477LDMmTNNnbMrNwEAAABgIzVs2s2/eLMMrNwEAAAAAI7VeyCdw4M/L2HnrpEs92qvqrDBmmNeoYw3g46GyVmfpmZ40g3F6Ei3Ss0tcCtbhw13nv6GxFb6W1rrkz/DF/dbYvv/Duz76x4x1h//
vza9+9avccst6f5Q/bt27OV7sFLN/ZbPewRkCf6MKn9v2v38aqWsz+ufL6amnkdaTprzSKA3N1pXpV++dx1dq/8Qe/FhrakxsT3HE5Rq8GcovVxamAl89uVqsgJpVnP85aTPplS/4vSlqqq3S1fQ2lLSZjVk1Kr5qWRVhsJ5Tqq
zlH1SiRqUJrrJUjy0/sm4D
evdwpZYMHgGqR3DTAbWAA4YBbTgeZAUxQdv2HpasD35vCHLtuMMgDd4ccRYQz9Cz5E56G4Y2PyUZmd
dHKDkM7SEl9vXGwRH6wRNz09vPS7d5rN2TZOZDcQk7tVWCL3Fdb3Ay6M3bydXc0Dq5FXkT8PbnIbZG3cp4j6AbXnqvbzWpuMO+VnPAq7nXCEmtVG
lb4Rno6yCHNIqM84f9WPQgPn6vAvfF2AeckM9Nm3J6WaeDmtR3X/FHS2p/5k7MC0qbdpzQkjMAcwjDBTbQxeKkjgpXd+dgqf9mZzVI77sYdU
5LLaGKHq7dSa+wJhrrX6nGPTZVukdXdiSUOG/gOjVC4PbGaL4trHOgYBRz4ZgJ/wDW/GvsRgQqTLP9C9+8ZsCLd/YTw6ZmUeOzyz6LMfqS5E
ebpN7Bwx67G/N9Ue9HaBGftE6TGYYIAoQYptA+Jl3qDZkPnnxQKP6pdttgV1VXuwcajY6zd5WYcOMjN3+xs3RR/fH2ZDN5zSr5QBQI0mfnTHPOzhYi
Dfc0P5ohuzv6SOG/VnTUKDdZsqL+o8ao8WS6m2GexIP6KmqbBwyHu/wK7Ytb6Q1vlyQDqvcnWFTYpdPfxKkLXd2SbDQQ0HgFvGLiJkKfXr6/Tz
+PAQ62DMY66W1l73h7mj0ImHWHXMxvpHFzCUlEuYPBnXt/+qKzqwz9gQ9fkFOMKvFJZVlUZ/lKGdvoBMDjm+JdanmYTN89gPedAgNfwrXLEcqa
GWvHZsssRi3Mu5xWUdGWcFrYlrxbX9l++rp5+qG3qkXDKYZGpxtB9aYjoNmIWUpTQFmATY2rW3AlnCoZRBGnfTm1ywR0xSBbhhxTgG6vWnmYjZ7HW
5nmZ6UFHSYpfumaZczjqbnq7dWA9KUQb8sbQT1HFgS9x81PNAEIRvrMQ6lTxx+AAgMy8N1zpMMwbG1hyJ1bV/1UWZnnAWhjgXWN9b1fNAgIAY4OvsU

wdZd5MAF/X
WNmi7W1akTN4rfgdJPw7b7N3cCBqPaIbJUxjgFCSdsj/QE+g3n4K0
IMAbhxwvDxuhAMLWMIn8Jl1BbHBjoGMlH9x8OQnw68ouj1eqR3JFgBT/xGyz+aN2hg/yMNMbi6IaIUsp+uNhlWgHQMYHVT3CdJZxBmDJCoCkH/5Z+q+IKMmHK/h3tqAm7Nc
+pjT7T6iQPkL7NtTmMdxpCz8xbn8CP4yXH9ftVp1aW9p+pMLrBnKmX+K4UhORWdIIE6uSyuc4p7H6XPxGvuzzAA09sEvo6ufuSlRMUFgWPIHuWgoTaPVq5RVnNyZVGlHmiMvlaOzO0oqWkPnYN69
0aIYkmlG6c6LuRlNaQrFTPh

rk2SVVZpz585dZTXn+xk0aFBqa2vzz3/+M//5z38ycuTIvPLKK+nates651duAgAAAMBGplQqZejQobntttty773
3rlI0du3aNdXV1Zk2bVp5bOnSpampqUmvXr0+0GtWVVVls802y09/+tO0bt06/fv3X6d9SFwtHQAAAAA2OkOGDMm
UKVNyxx13pLKysrxCs3379mnTpk0qKioyfPjwjB49Ot26nU3bt0yevTotG3bNoMGDSo/T21tbWpra/Pcc88lSWb
NmpXKysp06dKlfHGiCRMmpFevXtlss80ybdq0nHnmmbnooovSoUOHdd4P5SYAAAAAbGSuvPLKJEm/fv3qjE+cODH
HHntskuSss87KokWLMnjw4MybyvL8qO8666KqNGjSrf79OnzyrP88gjj+T888/PG2+8kY9//OP8/wuPnpAPbQ2QZP
58Y9/nK997WvrZT8qSqVSab08EwAAAAAsBSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAAKCQlJsAAAAAQCEpNwGA9eaJJ57IcccdWh0+Bb8
169bZbLPN8qlPfSpjxozJq6++Wp7Xr1+/Xr3x/4cKFGTlyZO6///6GD7O0Jk2alIqKilRUVNSbvVQZccdd0x
FRUWd/V0ftt9++/+xx77LHl+/fff/9qczSk0047LRUVFTnssMPec97DDz+cL37i9l6663TsmXLbL311jn66KMzY8a
MVeaufJ8fffRL/AIS6mz4CAOvFT37yk/s1eyCAuw+Ohu0GGPAmMWAWUHwWY4+DW4bHzXxuWUdt8+5NRyo9EtZ/g
HHHntskuSss87KokWLMnjw4MybyvL8qO8666KqNGjSrf79OnzyrP88gjj+T888/PG2+8kY9//OP8/wuPnpAPbQ2QZP
f+tKX8vnPfz6DBw/OHnvskb322qMhbq2uhdCyhb6+LRNTPK0bbG+LMp8x3SN1fy/y5f95+8rc+Jrt56m0pLi7ngWr6
LL7wwt912W5555pm0adkv+zUXX+ysedqjK7oqBOuBpTTJs9h7TR21VMXP+GPXBxmWmLjqT+1mGNxJdY0tyEH6S2WD+k
33313XnzxWy55ZY58sgvzNmcZs2jl8+NOt07ySSHHHEEERk/fnw6dOi3zvvtZEgMSs6hGqxtUr5d5rk9AsM5Gjx6diog
KXH311XWKzZVatmyZVkVBM9MlT7e6XmyeR/nOSbvYM9MM9ncHH6m/q8iaqEHKnjg1/3jizLu6XzvkLu5FkFfeFyIQP
PY06SP//5z6moKh3ReVvf/vbFqPcxS5cf1IN8wHRJzOryHDtHDJlZ2ZUlApg8FPqmoTHHmK0evGXlR1mrEP++ejACg0
dO2afffbJHXfcfcsprVEfSnuhV8Ed8AM/bM+6EujdRy/TMX+e4Yy6xx8tJgNBsIWzMBPr/7vmr/Bhz9gHSsJVSEGwd5mK
+5u8fZ26SPPfZYpfBJkgkT3nZemK9qcvc4K/eGQuEvSycA6aq7I2oiIVWV6+l6UbmNthxrxFkBDXNnWfUCFt+KU3LX
Mz6+9dRZ0bbH/vVW/5tEV6sMHkHrnnxjMqbBxeWHWuG8LGgkH7IjPoypzXfY7Kaf1Vk2Ad2C/Lm6pIfZ5eTFc13xu3
v+WgHX2M4yi5/4c5kWMFkySBLqwbsRx5z1BlFwkOhNSpeFnsJzVUzyycdrqnfNzQ/vifL2H7cAccsdnfdW+WdVcqmxp
3GqS+ufSPw3UDlbXpYbffmk5iGzoDX3LMSx3gHnMQoXzO3u+YVWjpwkaHjeyYKAW2hW62LMkbVvAmAfEY8kxsQRpjwL+
TvRGuLmmXTDWNN+JRbtDRJxvOh5P3R3ytg56fEDQg1WBLhvcyqDDh8H6N4fqm2kNOl4t4UDmeMO3vl9YpPoUPCAMo
IHjsWeDzKYdRdc2AuMJlsl3RgGqRGQs+b1V9u4DRz+f34DQzW2Ixuca+i7yrzQiAkc6NR8VXH1+RqXDC6egNLdPPA7i

oqMhxxx2XyZMn56qrrsrHPvax1R4S/K9//SuvvfZaWrZsmRYtWtS51dbWlg8ZX3lY/LtXTTZv3jxbbLHF+2YaNGh
QJkyYkBNOOCG/+93v8sgjj2TGjBnZaqut6hz6vtK7n3Pl+1vf3He69957M3v27Hzxi1/M66+/ntdeey2vvfZajj7
66CxcuLB8LtI1/Vr5xz/+kTZt2qzRPtanvlWm1dXV5ffzX//6V5K3V7a++/2/+OKLUyqV8uqrr9Z5/NZbb73Gr3Gr3/
KKafk97//fZ599tksW7YsP/nJT3LUUUeVc/3nP//J3EVlaUP1iiuSEMJKVJHBqK2TVyHiepdbb7i3V48fzCrFSeJrtbb4rMJk6aUA
wh3Jxvnjx4lVee21yAgCsNHTo0DzxxBP1nuv+3R84l0qlej+Efv3113PooYdml112yfnnn/+ez/Fez7OAKy
TZs2a5YADDshvf/vbPvSSS6s9l+K6OPnkk3PRRRfluuuy+LFi7N8+fJ861vfWqPHHHHHnfccfn5z3+eadOmpUuXLpk
xY0auvPPLKOnO23HLLjBs3LuPGjcsLL7GOoplNTvUMP9uN+20anj+D1HEOkH1l+6F0fdqIX+ecNUvDKOfe6KYNYdkvv4a32cV7Vvf
cKw9lX1nu1dbW5iMf+Uh5+/Lly9/h47TC+/iSLnIeBjx47N2LFj691+0knpVm
zZtl///3f82vlpZdeysyZM3PQQQd94DzPtqkOnbUvdC9yP0lRGzVNAcN94hg2Pq48Dt29+TNlTzzDI9PXvz/xlfl3GApzxzZRqPv4OeXoq/bOmTPbxlg8l5Uz
8ufvsN/IEzOIDnmfkm/xcX6v2LdZ0XFJmaR//RMvabVW3Gxum4aXV4XZ7uynrHmY9ifi1fAGAbSc8ivkl8E
gkyCfR+s1oHgW2ATgw2FpJMCJ1KZkO6S/Qtr8bGNnkU8bMhLS8/6KyugAKgpgUNrA/SvvG4PDUQ2/jUYZ7H0qtUj
ddDTAvs4d2YLUtN4Yk62K2gtyp5+XQM2q7R9UN7JNuvtoZIJJJoIYmQfdTBFuyXZaa1kyWmdOTL5tL2/xNw7/Ldt5CcKzbXNF9UlrGaZ2tQBZ4y9zB7wfuN7duf6GBa0+VItiNCDF
4S4d27Zl9TsaTS28VcmNqpuIfVN0pPO4/GydWrCdYdBEgFrRpWz4tWxNZM3sTcq7mtyYGKwUQVWCxE07MZq2wRMZjPFfzqnSpCfcXZuCMDkOc2epoHuXiwcV/eoK/lDsUTOiM3TYi0dKbG/AFXOrZ8lmUb22GXdDw09Amjmp+B7klj23zvVsT6rYB2iasXR1gfOsv/2UxWzr0TSLnY7mGW9uZbUWYprC5idNu9cU5bDo+nR3tBJTiXcC5FWgbqZYjs52wd9LZa+W4wgaAA4c1Ccj+UjPFTDNBjU2gk9ZDXqZVfRhTLG15zkprt4RrybGNzhq/Lj6EHgMY06HsmdSAkPtvpPjuVWWrlqRDFTjnWy5v3mPDDrTNnqn+NAd7RfN/xhn6uwHbSWkvYYOdsHJ2hlx6BR1dqF0qpfXmt6u/OpjQwqPLlr/GgswF3Tk+hrnaoGVOMJ9DZ8UF1+YAaeHXLoBjNxV3Wi9uNhLutxr3Kx1y8/4+OAvMwbIN0t2DpTJDtu9LNrIQ5cIjZoXSCjT1rqHDHNt7NAFEmXo8FJVm80ub0uZwVeYMLm4PFCE2hA0GAGGmhkqrDGYDYhjqJPi7/eXEKRZlZgY5S/c3kcbtPXMcI11DjnMhMCZ45XZa6IO8T+2/mEQxSe0y1YB9MLdGn+8n+X34Z5TVL/5DF/ZfScN5yvwqeMDK5nntl0WUJT3lYhp3f5/LrYYX/sp/Y4sx2tpa8C/jOpw8/yPpyVhNS9tYUU7GdB00x3w2YD5I7dT5dlN32jzxxUM1DNlk6fKpEt/BMcsSrSSQUMvQS8ga8gmGP/i7L35w4YHSKKc6fHKVMOxVXvhxp5/uqG3IQEwIXPGZI4ZHAQ5VGOajZpAV66FvVG8TlLyf4LLJU0PduW9Lm7xF1krbAN9DMfJH38e3lq2sVDVYVOK+9+Y7rnkq29Hqv/+ZxvxedLcDfnEzBWwWr+PiRlfjnXzu+SqWYsHJtc9rp2ndtV7lzvzx1f5vz4v9xtsvnz6/56nJGb9x53jzXvl1tnfcp+/XXz7n6GgBFvsPkoJbKsTdLWdjqy73q2dv/b6+lrs+gX+22T2+8bFVPqPDZ/85/96a5rd9d85RZ/tpZlZ+nJ6/4vOL5z/6yvy5hs+v+76u2DEv2dwMftDDP9vRmHdl3mfL/V6c+0jTSDhEiYnNQ5S30pnIvjSPfVLJ8VZvWcaNmiy1fvZ1UjmJBk1PpPvqKtqH3L2rHHD/TwzFYXnWGYy1RKmq7zcYhWvOvJB2rJJ7f5L+O4ZVr1/X9qmvX8m9uS/qQ3TeXyrd6AdBBo/XX3cR6fX/f3VHXyMrd73PXKXW6W8KUXLXkOiGKtVSg2kvp2jgzAvDrtuxWkCaK+bvh1/cmjL9bdp0usD8/Hg4Dgv+ne6WTSJHqGXVtJPeyk3MnRHZMaH8TeD9yw6+JEX/nS5t2n6SEUUxkcKf7sHx7TR+l3/d7Tlmjq2NR6cQMttlBl79G7O0YrX1O6WbHcc9/5HeRZ2NHqHKj8K8TIg6N5+ivBhpUmNzsk+NzzYxmHWaLnRoC5lfXpuaZwffx2GB3tNZy9B73UFx3ThCZZmQGKBVG3aNJOQUO6zT6LuQe6pnfv20a9+xZMt5V1UZnwZCDqOTYtFdupOWfMUL+8Ru0Xq3BfCyHkoDQqSfbMbbPAfAEgv6CYDOMVu+5xC9ntt72a91ALpvAIwE2LlrmD8GK5kBNg7JSPYEJDsLhKMcZSbPv3S2dPDAqRzYjM3qHCRlsaLtIhIwEj5r0mljzR+aUT2F1EzRIxiBlzjkPPVvZlGzmNSv2SkfUE/4pVOx4bM12/Yqe3gY6uOp0SL1L9/2wtdUG/6T2gbIbV/AcTLtEH3hvg1/4Y93Oho3ZdDqKHK0MLkhgO9Oxox9YsvbG03L0kDLQHcd6LqyGx4Dmoy5ceI+Kd9bORVv9oWuLjmxMmfR6cOvuLeVHjQeuRLdHNYPD8G8A/LrlHZJBeH3rjvLlQbZoz+7cjS9NLsfxc0ZvaGTOR6AzXzTSBBS1jo0GVA2Rzzd1zeO8TFMnDn1kQBO/Iplp7a/MgKMEpKbhaxDBGDSlWCJaJPFaAtc8q+09lVyl7nk58FnaaVXgyd3/Nz1PYzj/pVW8mbZDgm9+PmMQuwZf7N3f/ON9gnqhDDaX/RTl1DeS9VnvOLRL+/ySJwbi2oBcyl0dbdk5SnZeAnSaTmRv5QtmzyNPAPYmLgYOeWbKuQi0xPLU/3Atn9+wJdM8U+xqNuKkwgn4uPi3v6iGRfPVo2/eGcXMS9+AFfxDuN+x8Bb0/9W3WXPXqnhPW8qyZYYN2VN3WB5Zgqhp+SD/yRM/VLtgBvkBeYYwl9EbA+Kbzjld+cOn/hX+eefzzrZLH3L8Nqzf6VTt7jY38jCxgmS+ZdJmjmOKVkj2fVvb+XPt3MmgT+TXlCfa9lVv2zlt/QtGdJMUd5sAOpwtvfyY9LzbUiOqOO0yyv+WaKhAm7FuPZqCyU2uGv73Bnz8c5ehe8uIoSqTVkr8zh5uKt9VNQwXbiWUwSEgsLu4bOF2QyLjEVpKpx1ZXY1vUqV6knMMm+AAvb0pAcK7e2a5/Pt74cPAVfBiTkoNyEp+V0i3vbZ9U9FsTI4f3zbYA6eaT1kOvbLjCubjvYV9/K2pqx79Tdu5q6d7OZcu6txtAcPhDl8elW9h3C02t5dmcJb2WfMbkHQDrbqaQqAdvGo/WhjCrFOr2yNNAy+bLpY3snk8lQuBzZ1dkEpEd/wKVuqiHvTRlscKR7eaH6WuLPn2KLxFs8FX+jDe4+8OdCBUEgtdTk8HWySeE5dY9Jxe03ifAPNThm4ea6QcExvXaL/3EvLQFlDmRZNj3dQBJDqNbk7ygqDzNBzj3+/GQQE78oXZrj8gZd2vrxnjZwEUNFlNWvmuVjQPMPFTs3FjXzdCjT1qtwyyy+7Xq3x+XA5bMEe7bXiHE6NLYr2dGZfSUgBw2/4p8Kt6q6tHR5iC8eVEOOBwG03T1DEAQM+k+9+k9KUTtc30B5/Lk+RpXtAr77h37BsffJbIuzDhDOQTnqlS30bdt0mBP9jEOjgmr0Bqs/EcmJ+0ZUXT4J8chGDK/fjdwLOq2L5dVYvaFQfu3ODjX9a5ax2/dQwEzyEK/uxp3LkgZzNUPHqnvAf1z/DhkMszPZdGYbcDTvOTXvz0OFEtAc6SZB/NSD/NuUzdlztGCSrEGQfnOvf5y8LZgIU8GPxV3aGDaNCnkHsbcN/UlG39pOYlsWLt57J4o1Tw4n2UxCQU4F6L1v/dO4/TGG+oHHBK8cfSYnZ7ZcQuHeUfN6R4WjjC97YxdzRF30pPBz9XJdesruJCnqCCqhK7kHkR1HbEmN8hetQ7RqxMYvV+Q2L0fh1jD+/j5HqSUqLHLZL5xt/lq3XEpZzQmXVYmDpsRu+O3Ivq39ZRbQ5uUJrE6m2tG5W93tWdlS0fmxWGTFvrDkccgbylvaoPxlRlM8pcT6OdlXTJHRZVhRSVLWqgo/TWMxsbm/OKDZEwgX3TDA2vvtdVzfqNLgACf0X7u0bxR9gL1xRjpGhAVG7mnJ3m/qTMGXFxQGQTZuW1sgeJjm+V48vLJYkG6V8hbtnlfsQJfaPDHWPAMnsLzLkS3/0ZVE6BsrveV2cDYUh+ylPeC+z9hNt4vk0Ey69DYHyXzWuDpC/KFZ+31/2Vrd/SBN0OeMdbvPPSiOxFT3gydlePG+bdW3zEqhXyiGjMP4RqC2vPa7Z4HtuJEdVdVnAGTIOVkYn5/IAbvaY1kB3ly3q2vKFI5T/bXSa9hi37Cv2xJu5CLiKyVXwIKaEyhZq4ASvB1Ymod4AJbf1pe2Iwm5uXtetkxWjvGCiXa4z6RjzMa98IVNaHHRDp/Bek6VpEXB4lkSQ4Pe5lKSnXpUhGmHSB0hFsf6IQ1QwLBcZsbAEr2i2akJ4MRY4V5oq3Q1OXV6h5ZvjAt8+c7Av2tPi7Y9Z11HBAMn3kdiWe+zwLhuD5i+rhXWWkWd+rQ+k8hwyTwaTrwJvPemnV5W9/IYLK1A37yl1U7Yer6HvJvEWc/HjKS+17lz6TH+cFEiuC/Td09t/8Q4Om5DXVdk/9ZVtsybDpkkbV9EvMb+65YUvOYHfTn0yxNZOwKzWvnZQjAqUhZWmTK7q9EMpR3T8dCVK/3HjxWJMYW7dNfOaZGmXE+GPuHlpKkhumEHlDd3Jtr3WgTUqnHb7sdz7iYfp1H+dN2WubiuaAf2vtIu3nPZHN/i+Xbc+NWsTAm7UBGhAnfWVXtgK+ERQeC4SjM3r5RjzVnNKP1Jq8A2/mvoWngV/
9e3Z5scJNN92UU045pXzl9SOOOCITJkxYL/tRUVp5OUoAgCbsiSeeKF9RHHcmzFwhTZ5bhM/8ddMgM1s+o2sevmaV4lBJvQ1EzPJgY+XlmtDcDKE9wDp5wvjWyqZVHOizp0MG5rejXxBk4s5GBSnV0eLQT1oGBOS3wZ+cGLazoHfn/YyGTsJtGoDUwumSGXdxAEm+9h1SMoZTJ+H0Z3sx0bjFhXyLsZxnRDrpbB3hyMLbsQxSYdHx5ipQyYFMNKaYk6uHgXBVZCdeXOqRtDu0NMs2qpe9Jw1XbBpgRtm9IFvqzyWPZ3SDxPNR7FDWS1jqKAjNiOGKdpyLmcBKRGD1Kv+RiyfvqMmX1y/BWsuACgQhprzACigN7RXfhGmwjyEI0OUTOc+q9bydwO/NrqUU6o4MQf0QyBa4e7l2+k1QZEzAu5QBFp+ZTh47rmtAWh+LeQw5D72XHkqVsNBzkovMVBFXtk7R8+7nFdgK+SZQI7mb+/mfvnEFBNnP7EFWE4E6Dlzt0kYDLHDOXpvjhU+SAnEHTFa8Q73mlx+W7yYQHHbxFDWT4s7mDt5hAtI5dRG7LzLcVAtt2KCrwZZBEymYufyhK8qbwvH1ZPQYXkSLcIuKrJvuGhRunBm8W20KrOmh05m/rBMbBSy+sGkeVd08q2JRvUlZnC13AULTOtPjjl/sv/vRPtxYWk4DAzd6y3HWnqs//jW3+tKaENWfTT8+hFbw2jhs5kd3TBDiyM7V/cGONrx9E8f0vuTVelTnjTy9P+gL/ddWDvh6zn5qxwrVv5qZDajOVlcNqEbQ/mqaBmYHZPFtPzu/dCtvkAWC8kEtnmwIAWkb/sq0rCOMXS7QKcavwYbZQfm4sTJZ3VIhOkXOw7yhTTwevxxQxSD62nXfDqm/rBV5m0DRxAXZw8Ly7OOAg2NQd90Dcn6wD/n/NR4e3jqbJ/vhzfdFLkGnSKA0j/XqE1Hzhu7snSzlazWSFVsvqJ8XdKiH3pWpABWsN5GgCnB8P3LSGjjp12ZFBnMUuhNnEq3BVu62T9Zu3YQa5g/CB3AHvGQeA4e9eNGaHoSNaWDXADmnn1NbEpqWLArHU/aa+A6CLIBsWZVIVEtDyBOYfG7WHMpWASWt2Q2f6dOg6hlrV5Xd3dujrm8Ya1iR5c6E7yRTYzm7f5fF19q8TkUW5pAThQB1/mNHqpKUGIAXtEEJXaTxZjEr8jP32fUXQ+8PNrNaX1PKlXXGwI6SR3E4vbvDoBAAAAAMO3Ni3ZIi1Vd8YZ8r98+f9KyxXHDqkMjVqrAAAAAFCy4hJ6UkNh/7lklPkpmkQBNjYuJ7pvoV+BMRCWj+Nh7OlyXZRPnAAeMUVyAvHGIWm9tw2QA3Mr+C9xDW8VTZ5eHgbbx+OFtyOjm9Sdjpp3V/vd88zyB/Kbvdp9ftM0mqrZ/wZLv3v0enn71bIPpMeGYvHZ29UnnUzQr0Nz2Q0EAAAAD7Z0Lzp1EhYSV4xxV2A81KBXPl/l2zQPm0LzfAMfa4lrK0XA8K9c4S61Av8aZ7UTyAZ2bJ20GtAwJD35PbiZNDX3AWrYU9vX+kbHsvYoULI3/i2p84tpOdsZyKhjQEPtPQmI6G76dVPwEoyaf+iWXQKhz7H3WmmA6ihyWUP8ufkpywIu8oY5dHzH90PS+DKrBWb+m5RQl6awOaSDYKgN1Mu3+FYRPBQnS8tkoKoS1Ln8Q0C9t4ZKmxXEDeNf4UUUgnT3bF2gD0/kHk1bIBCmgwUm6QZBiQrnA1SZQjyqDh5GOl+UzrmbsO+Un8VBGrbqgh8dZD6bzDNTmdVMd0SXM9t8S2j6lyQMeFPJ4kfk2h/fo2/OHKU7ZBeFp2wKIbl4vuBKuCZ4s31VMiT3oumOuzuMdd8lFPqOH8cmaCoDzKI/NqfRfdB1V3xu7zCRKHG55U6o9T3ISfp7F1kkxIjmMo/DiP5+uB/FsFNLvnbfM3VFqunVq8yeGTl4uJr0OQFr2Ru9PdcUVuTqC5JyrvXiDK9uXnntqGqAFRPtHZoc2qsFukGpnvmuiEJkUb5ICgjY1Kd2PLX1uTvuuDfKJWXuBCiSK0KNgsrsmMLjRZfu9qzZKy9Mr2DWHYDBNZ7Yyp2ug1ez0XHXHIUnXKZtBXlQU7zXyCWGwVjlpnyDjeMdyEewuTcnsLfBqsz4C8i6oWM+xtBm7HHLHDTrXHBUKlvdN7oPBDPaIWtqz4/9zs6lJnYq+rhMRnGgZJoEEuDzWbhcl5VzVWfFDVGmhr3Q3fV2ze0/TJ1XtUqv5n/zqbMWLRSvNUXmJa1j/TwnZnJN9bdYE8DNxcY4uhHCHeEzz1TJV9kWmJ/s0WCPKs/WHl7oOSsOPHUWPeSdmI7qjhAdUY1bnnNOz5mJ2CX9eTqqgPzF5vwVhcO6bBI1GPvhTgjNgW+Ta3/jMz25yGLJmavz4LlvNtm0X3P4X6bjqfZnlurV8hpWHHJumT2lKb95bAT+i0RLOl+0tJTn+nKgnRZENDDgdnvvUSfUr+6Ha2ghAuU4AufOR/Dnp/ZA8Qn/d48KjPRkJqyyD8hsjjt6KFr0uZ22Ol6Q+3pBNthNMFE0+Tx1iGnd8oavXKu+zuzt2Jm/bnvbJ2b73rvP8/T5jutOr7/+XEoX6r8EX/NVQlNR3mv+lNc+HPZ0wmthhnbI9vDH9I2c2fN7qlkXOf1Un+nDVi/BF3TVJXlSnE1yAFhCi+fxrS17vN6Y6t2A6FF6u/7KzWY0mP6U20lXWzSRvUrkRZLaohcZ/ztlAQQIKZ1E+UKtRs5L67wblLsVSDvvUj4I5WnfkTKQrozOoZU6FDxCnJp/pdz6gWUm1o6mxQRgcrbXMe5AnWnFuqnTa2jFb3F9aJH7Tn0xuBy9JdF2WgmTF9s9Y9N+IwzeBr0AK72JwzGKdbwMXd7rzb5ggX2/PEc4SAPUoa1mDFRXxLA3bPQtWYywnHI6eBDuJxo3eSr8d4qR4d3VOLhgY2IiX5HAG3afCHg2Vb4jl06J94mtqSJFCfXcmTrjlSl80/K8xZyaHiTzOkUtRkUsTk9vcSCCHRP2U2Nb+B+DqrkkUzZThVhO+rgtu/mNRW3Ay9uwCtJLYk+vVWkd+G5zvuXhWK5ex2jvndFu2qk9PbqZYsGvT6mUx1QSudbUn8BmMvoYlhIFrQx/ekumIOdZc7yGh+Ot0Xb1DrGSN3Xcyvn5jNmQ+rsJIDkxvaMH+4cZWzhpqoHswSmbTtp0ssmSZKcimIx/TvhBR6eMz3XIiUgWxJXRPEtEVhnbkPgPGsU7EbnyEbbrLj5Sj1BTLNKBnsWi/SWmHm5okAQ0tMtI2p8FeBTe9bNvc0c9zM1HTJ0+Cl3FcknQ5AanjnCjxbMFOEfyFXAIgmhhWpxdJ9HF2ysmEh+gK8Kgf1f5bmDOzxhYj/u1pkwpdPtdfQ1cphctw9U0sNgY7H0O82DSJ5BIJa+tHSKwLuTX2s4Pl+Bl4fI1RH02BjqR2o3OYcVaILk0bLWdOc1F6xH/v1ZC7t1R9vH6X+wC1Y9EmJlxAXrYxCJf2fE5V9HVcGkGpWQSEYp+ptZ1ipzCpfPeR4ts5JgS5EglCMmW6kSKDgsFXLB2ZRJaGKrMoL1t1WlKFQ1ug4ViSpdyi2WyV1bkjkFXmz6RSR8nCUvjIAMytUJGFjC/rzwnQqldMqbnIXwfjsfhY7xaQy5cHwqgl96NSDcHIHNsDRi9BaiCaQWjBvqyzVUFMxvFKRhXL0eBCBGJoSAtPZl8tWQFtMVbVW1RP6xfwk9vgNjJOX9a8DTSXRg19oGEMbAj1BU7AW5QxqUI+xBr4KRh5Y3q4HvyvzPhjG7s8D2SEKsWn+z2KFfoRKTGmYXqq4q7KhHrtcg9lA14WwUDy8tPgF7MbG9jkrRBaKSGYjuOi+14FcqQM6Cxx/D/CWr4Dzq4GSoVQdNDPYR7w56WApYB4OD/zKtjEv/CvtCdXQl/D1SrnEIo4V06dLNRJt8Wh4sd39L6KD1bMSCFuH5Q/Nk8DJK9a6D+zxV4CtvxgjZHmn2akeZMOMSJeXKB0qbj1YwqwfVP8CiNlGjeubvchj+Yk2JfO6j5ZEZ4EBrSqrLr1NiR0GZJ8pYrL3X/zQgDzmuwQ7tu4tT3+yYH+4GqkDG8yf5aQGU8SWrDlI5WsfyZkphc+GXMuUuAtxvncQwm1ZvdEZpGxqvUbtNaSQo2t1DpPHMrlIwDhZLT8MOvVqhuT+QAwg6nfWA5SyLAnYS4hIy0zNlXT+RH3Oc8YBsGeP3X9CMQEGxAuGsQgdqEKW/dL55SKbKqFoURwVSDFn8zs/cJqtZPo2+l/lP/oZEeXuNbnDCvFg2HEsUq5jNqHVtz8gcr9NPg+x+h9l1BeXTDJIVHpSRp0lNTtIEqLC1WNh9wh3XPPwkfU6y19hd4mvBE4xtuw5nhT5IAAI+nVyhavRjuOlyt0oqPN/JEc1+/wYyKR2i7VDvIQOCMc8Op1ClwgujDS4yV1MbjYg3wgBmYB0yM1vT9uYQmCI+SSXoXYd+2UTfQwO75vOqvj2Rwn4upZGrRg8edoQAPg+9y/3KgcJD4QNRHqIH4tQo3fLGAvMfnP96PWupnFYKzmbcbRaODvLvzlLV5q7Lnq/DXgXDsSi6VDTsvkJhUkGlE79SJhWvvEULxvWsMrvNxiIvZJZfJzXZ4ULeNvs0iJjijf9WSxufQVVWVFJxG2fhXXYnLTRMsoCBCGTWJYIFitnIKhpaYyrYJ8DzPe06xwLo6C07mgvHPqzS5jcn8obyz5Trl8Lpif5bDnUJdiTpkRSTYnAGr1kh/lKljF/eDLhbbd8Kp2h7/DDTm7BtSdjUAx05Wa4DjmRGuEqHXQGRJnGJ6zMDi05Fvaek5mSYEMNS5ga0KmahwZUZrcGOyXsz5YeGPGUe4/qzdxkkpQSqPpOnXXA80hfn24aF7tTtVdLkFsbMFrL5eU7nIHDAfhpNQtxPdsQ+E9X/SXoHJnNqkcLhpJZjTy6Mi9vQd/IyOx/dgBEc6pThGsO0qClFIdd7Xhg5+RMhVhh6pYaFuMmMqeqFAo9Tp0CXRfxh8ewAnYpYhVDvIxsNw==

Xb9tuu+163jMAAAAAYENr1HLz4IMPzhe+8IXstttuOfDAA/PrX/86yduHn69UUVFR5zGlUmmVsXd7vznnnHNO5s+
fX769+OKL67AXAAAAAEBjaPTD0t9p0003zW677Za//vWv5fNwvnsF5ty5c8urOaurq7N06dLMmzdvtXPq06pVq7R
r167ODQAAAAoliZVbi5ZsiT/93//l6233jpdu3ZNdXV1pk2bVt6+dOnS1NTUpFevXkmSHj16pEWLFnXmJkzJkzJ08
++WR5DgAAAADw4dSoV0s/44wzcvjhh6dLly6ZO3dufvCDH+T111/PMccck4qKigwfPjjyjR49Ot27d0q1bt4wePTp
t27bNoEGDkiTt27fP8ccfn9NPPPz1bbLFFObnbsmDPOOKN8mDsAAAAA8OHVqOXmSy+9lC9/+cv597//na222iqf/vS
n8/DDD2e77bZLkpx11llZtGhRBg8enHnz5qVnz565++U0ZfWX6Oyy67LM2bN8/RRx+dRYsW5YADDsikSZPrFm
zxtotAAAAAKABVJRKpVJjh2hsr7/+etq3b5/58+c7c7/yyYAAACwUarp07fe8b4P1DRwksalJyqWJnXOTQAAAACANaX
cBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNw
EAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQ
AAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAA
AAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAA
AAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAA
ACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjNGzsAAAAAAE3XhNPvqnd86KWHN3ASWJWVmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjqbuXmhRd
emIqKigwfPrw8ViqVMmLEiHTu3DmtW7dOnz598tRTT9X53OLFizN06NBsueWWadu2bY444oi89NJLDZweAAAAAICG0t1ziz2wKAAAAAAAIBDUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbk
JAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQk
AAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACql5Ywc
AAAAAKLLe43vXO/7QsIcaOAlsfKzcBAAAAAApJuQkAAAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAA
AQCEpNwEAAACAQlJuAgAAAACF1LyxAwAAAACw/r3wvd3qHe9y3gTgIbjpWbAAAAAEAhKTcBAAAAgEJq1HLzyiu
vzCc+8Ym0a9cu7dq1yz777JPf/va35e2lUikjR45M586d07XHdPZ06e2ZrdJTTTGbbbOcc8rWu8mmzSyLdq3HyNLWfmzkbyKl/ZFyXvfWKn8gbhrwYjBb72X/vAjg8g+WPcg3KRoR3WyxvcWpyv4L5RJtu3r+bNmo5AHw7x/VqRAoRIQHsKGtrvmmNoS1WsJ3sy8eYZxpQWtrS9nW+1Bi0BhcCZ+XKpCbgCJjRXnlbCk3C2ZbBaeYxZl/W9VNmXzrrj2Yaa7q/nNanOCp3dqBwPyIjc+SFlbPb5dYKLsWrxIBz1k3LHGlhd+6t6h7Mi42X24IaNxQ7Rf2ONBHmrIXtKl1rwBkXyXMm92XVTKtyCjBgtlb7nNFVWzBXYdU0sSqFyl+GR7EgBsG9hG9zR4UELtOPmaklN18nLhS7YKzLt9yQwLcHkUqAk4KM9z6eqH7hNhQ6/nWpA0e0XNIY8KfD7eAelFn2t7PkBRt9sVfgPL6mfhDJ3gQzTjTQdeJpWGE4pRkUZnLXDrvxlSqurr48B1hZX4SvphbU3qSqLSPORpy8N7CtxYkPcRVgjbxGE7pNYAkW4TSz85Jmfm0s0OUTXp+nNC20BfWRsDV48YdUvBiUr9gbSCV8jjQfzqCIsxwD9Yy8N0qzrdwF04PnJlUAZl1nrSb7k0dVFDqLH5q8r9lSz0o0CgxlKRxiqeGxa3DW7OsfLkW0cGI4Gr00UYLVHCz/Om8mk+pJRJhd3drbm8NFYKEEZSzkL3DcV5mh+UWxxbMR9R96gMvtykQRqUj0fPzh1fE2oVIg22l1mRqXq1Od9C6TWxT6uvQOm3gukuomWeYLQaETE8y6wDGydzTOxSBBiwDbsPiGX/cEP14R1dAL1tlsHpIx5Z2DxtEGY6fHnTPmFZPm9KSFJJbkcVFqTvNRyNABdnexj1R3ks+jFUfZfE6X9yG+xdfNODnv+eHu2/Kk6BdTGBYXKbYQW2JyoHkg9HAU4OaQy8Wa1XRmQS1jnpcnuBrGHW6s4EKDuPbPHeK2N9nWXSHYuixJKrr+IbwQDbh51rtzq+uBYBl5dBVKc3zHJ8TTxyLxBk2yHPFT0evB3P2J0J7vG/wlPuE7AMXZSHYs+6oq4Yb+Iv4NP4i2ayD8xEcNYR8dN8+2f+3TWb/k21DsCK7XNZyN5jWMVBxNf2jlU7U35Rr8s8NddfJgXWwkOjyM+fygSfXzr/w8Ggo7ZOnN3XDKyeXRuK2/+7mShNH5Lhkpf5Nxfk1yOuPgi2T0hkVL+jQqEtaFm86FT3KZP22yiwZTFu+6SjIIzLxbfrX+/LYXTbEqGuDCTrAHc9YaQ8g1kUwAsDDZHBfRFC0pr+M/6qfZd2Iqd6Ymp8B0f+f+Tkbfs3W9NbLBAzJmk7/bHNvP/vGh2I8LwTEe0bkxHqpqOU3dFRSUuGw1rFttNq4+3eT+ja4dJvHiUIuDFXvuwa4xX2F7qLsVr6ryftbdgVTRCbkDvK1ocHtGx7s8CrTbwG1LEYpwmLoyYKFi6aBAxNhZd9yf9hXF8sTk1WuFIlxeSmRdwVpFkZrzj11zuCPLVdULN2oeewwJVz9ipR0fzNJ8gkhPNfwo90gi/KWKwZUDmxgX9nsryLWhtvL/K8GQYp2vaeehNxH7bTfnPmLQQYY0d5SknYyrm/xAlOgVfFoM+0qc9iovJQ6kfYy27yPPKrfw9bIUNT7rE0dVhNBtvkw2zJZ3Dx9g/7lqWJE0TQ2MJTp1VxJ89VaNgM7fQuK2eZ9DxQBfdTvjO+A0Go0i9ywZIE9ZXw+Tqm39f/sQ71Tn1VRn+LNJjzJlxz8qM+Ri1oAGv/8Ht8GTaTDjiqxYBivcZwyCLbbHpGdrvdFcXJ1VzIhkj8KYwVVJGKhR9sI3GeJqEXflCk1r4qpX3OOzN6xtZYQlqhd3bJyNz0XzwWzd7Hy7fuW8z2ltgXVqy55Vc3FSPl3DvLWqp8TfOPbo2ZVXN/+NvH4ZcBUcE1Fn+eT2g1wGtZxBfgFgy5ZPIQm+PeVV3zg3RjWtfXV+Fg7wwV+PXbJwr/Pb/xYpevfT+QkKBNAZLCYsOqZzO6d8vqHaq3E/lMjIOrN9b5n3JTpKAdrvAB2tXbxJcoMJcWN1CDz6fDEtzBfFf91J8RxhbCNc9+N7HG8SMo5+FArYmE4nuS+Cmw5SKg3Nl+1vLCL9qazNHdyb8gVwdXFKtj3K+cjVgcCqfDHUGbVgPcBeFd8PM/5aMq4NvQgu5fjxEAmA8wQ86hyqBbyqzN1nzjj5+X1jBEP2zL5e4XX45H9yH8MoN15TAQUYYqyOO9P2kSZ9bZfBYNqRxJKmDkm/MHIoXTC8kuwgfZaLF6Em8+Rr7gzMXvbc8/nSS/ds2Y9hjEtN5yOsj6WLyx9TPPnrsQ8aGKjSRNkD1n1W/RhaUD7tDmjPSmxvBtUiGnZHGwdTP1AZu/uN+Czg8mS+aD/+6Yj3GhnW+ez4DfrdkJ8OkE2FSVD16a2sKZyx6CPcjPmRPfC+6GOJMfzrzL7YHqYeVdz5wa7XxwOGd8sPA/CjGzmp4eW2wbnLH3YAAQpt7e2jequ++NVAdfULySyQEsLR1lTLoqMj1X7uTUDpMZjbZJAqCvFAIlsAvF0cJ4DxGZnxU4qNuRJ+yWQJAAAAA8frGBebUVT6j1zlKPQbaSkHAvn1kEMOMG/JjxY6y+OGBl5lNeNHfKzCSIcrxX3Oa8kRg4JzW/nlHG2sY5nsN3WaHJ0T8KukgDmMQIjOtjA5jROYhhKZWKB5M7QXXKkCzzvZ2QqxMq5WU3NBb3L+vxM4VNsl5Ge4FJABklq4+CI3Mp7pvnbZCEh46jbCGBCxZEsX5sTczxMHRgnfPT2fhc+nCQjNBB5yz3GRhOqYWdQqzJVvw2Vc7f7P0nCqQQkYLYp/IzRr4o5S2Hjvmc0jsG7W6PctfkLbpcIWt/kMLEl2Ezc+qd0t9JTsBiaKxT0GjcSZR2fiwMmyV1vxNhq3LNOZKzJZGKk9aMmeUYbqOFSfy4ggqy+3+Ox4qgOt7ajDPxfUrwCxT7cXnVTSpcJNcHTvfsf8ouNUJwdxo1JIl0ZnPncnL5Wno6blP0WKsLhLPmxR/K2XlN0dPv+Uw7eM/SxzT2oNbbpF/bpb8VHbV+0jQfD5b51DmFRs0vWmVOUsvKnXdIdPzK3ZKvnPrxsYZCMphn4zf4B4mqf9dEkC+v2Zeu6xbRx88hM2UYgD2Js4zHeYMznnW+Q5Fh0ItXEs+CKN6FhaKDCXX+u8T4KgjCw9y1ohjrSd06OeWqeZVXEDg9R0OoL1Hf9hcfIIcDUFMNe3UG9uzl7PS7DYnW5uBNZSxzlWGgHDRA3JNj2kMemNo8/bVoZmHPQ6W8vCyTW8rGzh0SJRpDwOrSeRrm1Zn0XW9MlT8BqGyoL+WmDfdUBQhWk1dfXOVZRr0UifcvPwE/vq1ef7VzX0+QbVQtwzyg8v5fxu62UjZ0o1M6nNZkR+dWH3xtCzfwxqwEEiDuw1c2o+Lnm5eZvdo8Pd2fhjbqEfpfulsOC0zMYdwGwtsovJ+TdHb1EOcZ8RzyTpYyVRmHnF99CGiFB8e+kB+bwo7Yh1nRHcGgL7rPOqbOZr0/gbNhZgAmPTBD3rkMnH1EbxJvzbJ8Y/nWlkcJfNs9ZhGtKz+OV9vwhMplBaJWrdArlGA+VUsgNvdpBr9AnbnvQdo2uLXrGrsm0gk/2vdd7Nw07Zr7h/BLYaEhUydGwgnrXebQmQyTS49O6q+DzNzSGTFGW+Y/F3MOrdTLeRVUGy7Y6lwBnXI0vWfMiXn5xePAFlVjBBHt1u1W5dLAe19hkFXXjDc/9xCbY0MM+wTr4Xy9xFSrz+qQ79NDKg12P9VHhkN7cEgPsqSlyQDeQyhdnfsUQbttCGlFtp6K0QmsxHFaYmrjWlr/SrXlRw3WB9YRsN0xDfKGU0Qnq+nY2p1NvuwAtfYk00Ogye0j5wgyfBmRpz9bfHG6YzfB4JhCSPd1XcGYwL0hiJjXkIXvmw3X/EjSNR9b0gJSMfxq23+ww4l07OQlxO2XvHc3BL1p9zSK7WQYRvixl7Aw7PPdKzTEUBOgx0EjR6m4sU1HNrBV+KqG30IbE9vl+M0jzDMvTJ73wNL+3Zru2sVBWR51SRF3dlE5zCmq96Wei+o1HAfPJ4rB3hkRmW0wSHtRUr4SlZWxAubQAfklZJ7cMjgs8VzSmDcLl8zw9lvRW2d95gDOGWjxjvpdfvbYgKXl5NwYBmwb5c/JTRq4Fd2hnoz2EyuXxlvXWsHAq3U4eQi5azpUnM1yLhJOBF8D+PGCf+03XnDnK7GJDp/oI4JswhsdAXiiO6uJ+nNRCtq7sdHp3C86DoYqtOqZnwUtjGlx3a21fSTSNMd7ZuTYe6OlyPVBO20Eeb9/xs8P69GZXgCV1RgW1ETXL82Qj0xgYQ3PIXnKRsj0I5GlmbMp1o0rbXeYjJNNlqpjh7ee7SpO/wM5J9PwoJVEqUy6rpS5GSs8VIN2BwszXXRyoXp6MfNVqwnMi8utqV3jP5a3Q9vjGvVh6wYMsXI/4yHTbFOj2BXwNgEDx4rWQAnNz9JMGHQuJoXGxvuPSbD2Zm0Vlbd5PL+1z0TlC8GmRrx51pkwLvxUsWNJjnZRd5m/BZsvMz5tFc7G8xN8tRfwwrmvYm+q3c0oMPO0Gcf6nxBMwfgyc3wovM2Cq26kZoHkeLLTQ7IGGjRlLsvKA6t4tUtWmgGp1AzKx5k3vVP6LQqvNR3p5vAwXc7vLW+XA4CPKYB+M2mL8vLBnVOkxrRTdk/rdcvLwyW0g1s4tMDBMGsNu2EhCAyi6C6UJf8bmYnZuiKf+3Iijk5TpXJwDaYHd2s9Lda2zvzqzZPTgAEJyWI0bLH0/yOuFFrDMNHq0lk2/Q9VPt0fs1WxrYmvJYIm9PNEGKDBW5bAaiIAJsP7pFdwWCzNl7YBg7BXCPRGbkI/3AWPVsMLsEB8hLrlGkFOsDU0i3vu5H/VdQJntHeC8JxjQv5mcCoy3fK7B+7J8FhImDG8FWBrXN0Z2zRbZF/AjJdMLh8XRbPnRCDsyY7jUOfIEu7V7eHN+QxNlLDHTg9XzN3+ahIQuUk0Lw0uSd/SRbdzJW8SxQWf1nFKeWjIu4OfZC4kgZ+uZE4WD26LO7A6v48MpCkJ0ywdQnFyyFSaTJmLwaVXlYSYvSe0VeTR+qWn9vLGSNcZiE1gIaPiSXo1xglVeDq7CD2dtgi3kfOm04vr7+mAUfaK1qT+Ty3HcLZ6wWS4a2L0Yw/OkFzOuRkITaMT2sJbnIzfzJpwAakiyD1mZ1pyaxS7YqNoWxamtSDwSJ51qf9v8Tbb7UEcj7DXlzYRq9qtFXnB3m+/gfJZPcvTIlX0rGL64AM9n4wlO7LAMqZUXd9Ae1i3HAVBAA7eQPaJfY0LQy1gIA73Rgx8IbfEHVALyGxymzx5MmrpEn2FCUGXfgNIA4vV/bjNcQSpTGxFl5wOm2gM9lEzR4S1A8lw1qrOY/9lSrFbuBQJQpELfLd2pwYBzyK5x1nnT/ZBojc0A3MlAJQGN+vy/TY7RWLrWFWA3QqH4s13WZzHaxJFMnfDlKmTbkZYNNWLtzd4PnbzK/m/dzT0Rfux2IOOSq8aqMgwB0vpl4PRq9Ks2uMVMM2cHVZLLtnhIBT1A3ApgkZ2hKRE79B7VRH5BKKoCvgX8Bg0eD6YlYXbhKCgfMVwB5Qqx01ssbW0Xm0/H8aETXmKcS0Xd/1rSqYVZq7IJtDtYFOSIxLJvF8UGoPZhIsAqDKNIO8ve5AqHfP7vUwCx8HDdt6VuKrLFT5p47Ex5HsSTBmDK0xPGvLc85QZbLVbRYtHaBbgRdUAXBjtBPsc+w1FhJb8zeixYhc8pg1z1XNM0w9KoW6GCdF/37iPX8pB+uQTpSs4Q5x0xpqMXMoRBY4XvbXkPkRrXvPbgUBmKCP5gLcSD6ToITrXDJDmU7qCjMcGDDQcRm8I79+MQYRLPGUNAE1SVMM8ML5jWsJHdEAyLflaW4u5Yl8Fvc+J+LoZbNd8Gq45MrXb1iI6ppVcjcT08qqJHCoPW7X/KmPAd5MxTyu2nAm7FymGC4Z0o5c6kqjkc84DbHFUXOsMNJ/D86ib5s01jmNxKhzu4nMrQZjtAoLSrSIBlDlKVYPG2MDA4OoZKWNsXgZxUJXjw3oxzQ6/e3Pr3zUhnxmvcpk7fsMX8C1xvZLrMyAwx9AOMh1RQ8GtuzrkBoP3SxZhhzrZz+2Lz4oVXQS19aqyOuMnvUpNmZ8xQfMv8/GtXcx+8zxoHTgYUHL8iHIa+OmRydCsjMkLwlB8Kz9bw4YjfiZBbGqIH6n02UF2W+jb7EjSz6ftVl1PQWm3c7RqfhtGEYxs8k0DByJPcYXi7VmSPxAFJSzF1tdD5Mhw6DWjBc6pP+1L3kRzHUyeG3H+8P27mpP7kXGQ3OSmK8gSnX5jx/Xm0GbP6gvYSPQzxYz3XU26NAa8KlLGeWfWAHOdDrBOGRmM5LCFyeOD1KGbMjkzDvx1I9UU73hKiTYBm7R8tr6FOvHObLcPzLhP5rNAY+7Sh/wZKlZ8PA9Zdxo4sD7k5JK+JDQ+fjiTmUV8NgK8d4tyx1XfvKNzSqXbtbKyQVVw+YaObk8y4yT6o+yvnpPTe21RMLoysMrtx6/SLR+J9v3h5sW7X6D+L7TNBjhNx4gwvYBfP4RyGtz1tC5+uCmMl0qIU6RaxtzrSMbZoPvbDsC1hwZE/hN4gMSvvL2GWX6f7k2QkBXX9YhKDFcK3eKfcA3cdwwg4jq5I0VVo9b0aI8KYI+3wrLx5/z4k6bFtYb+C7pnJAXxs5bI0jOWIT3dN1AL6y3UzqdtmqYc0iiB3zbKSqLsz8YZ3TvvQZqD+ZeEs5zR33xkPh0L6bw4qgoZq6GtMJhJkmnzSbsF8Ni+Oe1ZoBdjjJPmd0Yz3kjjXp7M1SoOMR6xcHXlm3C1dbi/q3U/3QHHjXxWJ1CZjdgNdgp0AAAABAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

999wzF154YT75yU/mhz/8Yaqrq5NklRWYc+fOLa/mrK6uztKlSzNv3rzVzqlPq1atyldoX3kDAAAAAIql0cvNdyu
VSlmyZEm6du2a6urqTJs2rbxt6dKlqampSa9evZIkPXr0SIsWLerMmTNnTp588snyHAAAAADgw6l5Y774ueeem4M
PPjjbbrttFixYkFtuuSX3339/pk6dmoqKigwfPjyjR49Ot27d0q1bt4wePTpt27bNoEGDkiTt27fP8ccfn9PPPz1
bbLFFOnbsmDPOOOCO77bZbDjzwwMbcNQAAAABgA2vUcvNf//pXvva1r2XOnDlp3759PvGJT2Tq1Knp379379Knp379/kuSss87
KokWLMnjw4MybbNy89e/bM3XffncrcrKyvJzXHbZZWnevHmOPvroLFq0KAcccEAmTZqUZs2aNdZuaQQAAAAANoFHLzWu
vvfY9t1dUVGTkyJEZOXLkaue0bt0648Pz/jx49dzOgAAAACgKWty59wEAAAAAFgTjbpyEwAAAAABWeuF7u9U73uW
8WQ2chKKwchMAAAAAKCTlJgAAAABQSMpNAAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQk
AAAAFJJyEwAAAAAoJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAIBCUm4CAAAAAIWk3AQAAAAACkm5CQA
AAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAEAhNV+byXvssUcqUcqKired95jjz32gQMBAAAAAKyJtSo
3jzzzyyA0UAAAAAODq13t873rHHxr2UAMnoalZq3Lz/PT32vf/ny4j3dwsqrK9V5hwMPUssDBp+lcwgNRnYLMlsmCMMpayPWS2/IS2EhGxN0+eRWA0o9ryJeaoF
9OQYuqgsaYp11f0o5CfAHVRh4gsNNsAAAAA4P2sVbn5X//HH7H//NNNNBqPNWVa6HrTnoeGGuF8fYIaPFbZ8wg/AH7xG8aUqtYKOT9MWQrSfSgS6W7Lxm7YJKJbySRyhn/e1ctEYUUAHdQf
AAAAADq13t873rHHxr2UAMnoalZq3Lz/PPP31A5AAAAAADWylqVm+/0xBNP5C9/+UtatmyZj33b3sY/n
4xz++PnMBAAAAALyntS43H3nkkkRx//PF5+umnUyqVkqTfZEJPZE5+umnUyqVkiQVFRXZa6+9MmnSpHLJ+eqrr6Zjx47x5rNy0AAAAAdfQ484Z
6x2+vbOAg0AjW6mrpTz/9dA444IC0L0PLLueKKK3LFFVdsqMwAAAA
AAGt/zs3+/fvn1XX73PV9D322CNf/vKXl96hcwlWWCNf/vKXXPZwqqLxjsNmarN7+XXCFtagv73/RnB9XFi/5DbBsuDCmPk/fbDSMNqj
AAGt/zs3+/fvn1ltvrXPV9D322CNf/vKXX/t4hv2jzjm1+MtlBK4/sbN8Ut/7ftAFXxgv80Ly3E/tx0+7/j+Brg9zHUnOeOlvNmp6AzM8tcUR/hgO8x1PIJ
AAGt/zs3+/fvn1ltvrXPV9D322CNf/vKXM3DgwOy33558cUX89vf/na9hwUAAAAAWGmtys37778/999/fWD/fz/nF/6vsdBZwl39ZSpP3cGMGw1K/8/nKEb3bnAAi8OGcm7qs8i9aanwbXmvqmU
VVFTk3HPPPTc+ePfPb3/42ffv2XW8hAQAAAAAAewRxBAWbFTFXzBd/s3sQD1WqzaqpWu726ujotWrTIZz/72XUOBgAAAADwxtaq3NxbrHHAo
+++3zyCOPrHb7H/H//4R//4H/7tZf+y5fi+wnx7WUbv9K8+Uw6DTyrM4s8TxI3m8nynTGsDJbAqW0AAAvDZq3kjdwjLq+8qgLOKtL8wWiXYLVkqqL4z8sPYM/Ol6vhHpmmlZknVlbnGB22MXH7qXdmGRhBs1qsKGWO6qvB7i6h/Ovm
TADkhBJ67OupGMYXnN9iO6H4WOOAsaeaBHggxfV4wHXaFDfsM/kPawgQ
wXbWS5CzSTjKHVs45vzLONm65pnfWHM9LVU3zpfrQ9KNZzf7nnpvPqx6qNGutI7+tv7mkv71rekT4/rUrFKF+7j42O8xm
VZkOAbrm4ktPEUXBGc0odyjeLgrWPMVs1sp9trbCVeXIa+e7g8RdEjOmLLrNuWXHVQTr7wQnA4wMnW3Qa/FAmhGkMwY9Ta1Ug7v39dAvPUsfNW0t6hTQ2PtJJ
2p/84uD0dKtK+x5AMI7lTWXP8Rvuk4xxBa2QHbW5SwqkWNAWnfAUj4l7K0wSSS9WlbBSYa8cfs1OQW6w+pa7eLcVkpb9vPzfTHgTj40aYmQ4AAOOtRbf+wXPvgN6xKOV8dlmOMzzCOY2wf2/3AQOyB4
5Jy3C6uwLVzzOrq+VE3hlfNA9KpAUJwzN/3mDmUDuSMqQcC4qIrBRVcjQDcVrVLzj7KiLTXeGzHHX3z1bmzyE2cB/AcLmPyQXuVJxkQMUcpFFF6DA7HbSmaIPuz8pWWKz4V+rHTMAGs/dEu2RV5R8npSrQ0U0wXjaJNFwhU62+wh
eTI/ZeKvaCQC2m4VEOWVxP6mgcb9ZeTuXcsyMyY3jGIIaAhTp/Mm8Ho8xGO9tMJ4LJMaynUFq0lvn5qFVPw8qmoLTvxhyhdEPcEs3bK0jC/IBAh9uzZBBg8Z8lw/Jn/D+YbAMgUO0gFyMgT4BvOQ/w2gmE0Av7ncAdNpmu0VTO2CsbR+
j9XDkSr/yAMuWAzYwwsF+vwxmFyZh9k/+ALsFpGeh5QCFjoqfTGiQm5aT1eeR3TUVNmwRXw8vHNnPdv8KZGAHUmb6PJwX6QLf3sWzsYNqvMy9pvzWIAnN3UVuAhkKm1eFnmGvbA8PRPSbDGZVnomGuH+RZYi2Or97yC7MW0mnqWG
fUhHnnV1yqB2P1+55TGrW9fNAp9PjtEb5DJ39nyHzgIyGaMyR4U3xSpO2w7eF+4wOO5ucLVlznZBUKZR03/bd9xG9aUBA8EVk8cjegHPBG2CfgJHQfmQ57QJ7dR9s+TL9rS9RzXgtbS/KKhT1zUm77BeS1+4e6o3r41H/V+H4jZUpqj

AAAAAhaTcBAAAAAAAKSbkJAAAAABSSchMAAAAAKCTlJgAAAABQSK6WDgAAANAEuPo4rD0rNwEAAACAQrJyEwAAANh
gepx5Q73jt1c2cJC1UMTMsLGychMAAAAAKCQrNwEAACg05ykE2HhZuQkAAAAAFJKVm9AIavr0rXe87wM1DZwEAAD
g/VkdCzRVVm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJo3dgAAAAB4p5o
+fesd7/tATQMnAaCps3ITAAAAACgk5SYAAAAAUEgOSwcAAHiX3uN71zs+2p9Q1MNh9ACNx09mAGhgL3xvt3rHu5w
3q4GTAACN4YKvHlXv+IjJv2jgJADF57B0AAAAAKCQrNwEADYaqzvM9KFhDzVwEgAAYH2wchMAAAAAKCQrN6EJmXD
6XfWOD7308AZOAgAAAND0WbkJAAAAABSSchMAAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCG5WjoAQAHV9Ol
b73jfB2oaOAkAADQe5SYUwAVfPare8RGTf9HASQAAAACaDuUmAAAAbAATTr+rsSMAfOgpNwEAACgEZSEA7+aCQgA
AAABAIVm5CbAe9DjzhnrHZ17y9QZOAgBQHL3H9653fLQ/VQFYQ1ZuAgAAAACFpNwEAAAAAArJWn8AAADKLvjqUfW
Oj5j8iwZOAgDvT7kJ0AhWd36ph4Y91MBJgKbO+eigGFZ3Fe+hlx7ewEkAYOPit2KADeiF7+1W/4bN2zVsEAAAAPg
Qcs5NAAAAAKCQrNwENjoOGwPeraZP33rH+z5Q08BJAACAtaHcBABYDR+GAOvKxXkAYMNyWDoAAAAAUEhWbgIAHzo
u5gUAABuHRl25+cADD+Twww9P586dU1FRkV/+8pd1tpdKpYwcOTKdO3dOmzZt0q9fvzz11FiixZsiTDhg3Lllt
umU033TRHHHFEXnrppQbcCwAAAACgMTRqufnmm2/mk5/8ZCMmFDv9jFjxmTs2LGZMGFCZsyYkerq6vTv3z8LFiw
ozxk+fHhhuv/323HLLLXnwwQfzhtv5LDDDsuKFSsaajcAAAAAAgEbQqIelH3zwwTn44+/P317323HLLLLXnwwQfzhtv5LDDDsuKFSsaajcAAAAAgEbQqIelH3zwwTn44IPr3VYqlTlJu3LiMGDEiAwc
OTJJcf/31qaqqypQpU3LSSSdl/vz5ufbaa3PjjTfmwAMPTJJMnjw52267be6555589rOfbbB9AQAAAAAaVpM95+b
s2bNTW1ubAQMGlMnNdatWqzIrTuecbjq4Lsra3pUVvdZDbAbaAFMS7Mj4BwMrfsqhxbuNFfmTLZUGUEKL8vzBC
RACi41Z2/sst5sxo4CQAAQF1NttysralNKlRVVdUZr6qqyvPPP1+o7+e07Jly2y++eV7D/1TD7/Ye7jPP0zDOLD
uavr0XWWs7wM1jZAEAAAAmr5GPefmmqioKhzv1QqrTL2bu8355xzzxsN8+fPLtxdf8HG9ZAUAAAAAGk6TLTerq6u
TZJUVmHPnzi2v2v5qyurs7Spr7Spr7Spr2SpUszb9681c6pT6tWrdKuXbs6NwAAAACgWJrsyeLZPTp0tT
U1OTiiy9OkvTo0SMtWrTITItGnWTItGnGCgKWjUcvONN97
Ic889V74/e/bsPP744744+nYsWO6dOmr4Vu6deuW0aNHp23bthk0aFCSpH379hk0aFCSpH379hk0aFCSpH379hk0aFCSpH379hk0aFCSpH379hk0a
Hjh1zxhlnZLfdditfPR0AANanCaffVe/40EsPb+AkAAA0arn56KOPZr/99FY/99ivfP+2005IkxxxzrmZaSzzjorixY
tyuDBgzNv3rz07Nkzd999dyorK8uPMWbMmDRv3jwnnXRSFixYkAMOOCDXXXddmjVr1uj7AwAAAAA0jkatNvfr1y+lUm
mZx+vKqxYNTpRqq9CWc51cV/vfQu/S2Ez0aAIqA9VlAgWQAAAA0CMJnLTtKIBVN01NP1Rt/2tfzBKCHmZ
pVFrt9oqKiowcOTjIR45cqVUlW2nSvBOWvYYRP7ZEgnb4rlWaUyZGbmdKQh3ZTGVdUKxwMVMn1vJmB3zEmfFyLdzVQk3FdaD1x
AAABoCJs0dgAAAAAAAgA/Cyk0AaOJq+vStd3zWXmfUO26FLKwfq/u3l9X823P+SgCAhqfcBACg0azuNBYAALAm
AQJKkx5k31Dt+e2UDBwEAAFhDyk0AaCJ6j+9d7/joD8mPaxeaAgAA1rcPx19LbPRWd06svg/UNHASAAAAABqKq6U
DAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJo3dgAANh4XfPWoesdHTP5FAyc
BAADgw0C5CQCsVCSVzV9+ta/Ya8zGjbIBlTEor6ImQEA4P04LB0AAAAAAKCTlJgAAAABQSA5BwA+kN9L
AQAOxchMAAAAAKCRLKwAgyYTT76p3fOOi9lhzwko2PC90AAAAflHKTDzVlBbAxUhYCAAAbC+UmwEaox5k31Ds+85K
vN3ASAAAA+OCUmwCVfC93eod73LerAZO0nRYBQkAANB0KTwAAOMsT+59PH5KwEzW4y+OqeoAAHBKb2AQAAAACBMkjkc10PjyB03Q4wAsMVvQ8cbqEnVkaA+K
1ngEAAADezdXSAQAAAACoUm4CAAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACgk5SYAAAAAUEjKTQAAAACgkJo3dgAANh4XfPWoesdHTP5FAyc
BAADgw0C5CQCsVCSVzV9+ta/Ya8zGjbIBlTEor6ImQEA4P04LB0AAAAAAKCTlJgAAAABQSA5LBwA+kN9Fw/K
ZuQkAAAAAFJJyEwAAAAAoJIelA3X0Ht+73vHRvl0AAAAAAATYy2AjYgRSEAAADAhuOwdAAAAACgkJSbAAAAAEAhKTc
BAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSScxx88YOAI3hgq8eVe/4iMm/aOAkAAAAHxQVm4CAAAAAIWk3AQ
AAAAACkm5CQAAAAAUknITAAAAACgkFxRinbk4DwAAAACNwcpNAAAAAAKCQlJsAAAAAQCEpNwEAAAAAhaTcBAAAAAA
FpNwEAAAAAApJuQkAAAAAFLzxg5AcUw4/a7GjgAAAAAZVZuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwAAAAA
oJOUmAAAAAFBIyk0AAAAAoJCUmwAAAABAISk3AQAAAAIWk3AQAAAAACkm5CQAAAAAUknITAAAAACg
k5SYAAAAAUEjKTQAAAACgkJSbAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchMAAAAAKCT
lJgAAAABQSMpNAAAAAKCQlJsAAAAAQCE1b+wA1HXBV4+qd3zE5F80cBIAAAAAaNqs3AQAAAAAQSMpNAAAAAAAACkm5CQAAAAAUknI
TAAAAACgk5SYAAAAAUEjKTQAAAACgkJSbAAAAAEAhKTcBAAAAgEJSbgIAAAAAhaTcBAAAAAAKSbkJAAAAABSSchM
AAAAAKCTlJgAAAABQSMpNAAAAAKCQlJsAAAAAQCEpNwEAAACAQlJuAgAAAACFpNwEAAAAAApJuQkAAAAAFJJyEwA
AAAAoJOUmAAAAAFBIyk0AAAAAoJA+NOXmFVdcka5du6Z169bp0aNHJkyY0NiRAAAAAIB15ENRbt58880ZPnx4RowY
kUsueSS9OnTJ33+9a/HW//dfdf4KuHPXx+S7tqe1QTPHKLlUqdlKH+4//OOnLlbbbRO8Ev/zq+7ufvjX/5Q+uNe95r9+s/n21tkCzJ2G7TTWf9H
ADaQwpebS3D4zFXu9c5w7/PCnzaz2YF/TdVzydZN4qO/DfqN9j+5x88a4O3T
NDbAvvBxwRQx7Ja6MrbMTYsdv0nvqXvdx3lf8HnCCf/RuKUjd7NxaNX6YrPPd+nHaKDY94/aIzHfLk/qdJ+kcd
MwIcmKWS+PxbYAtKVmIpJLbyCrgxxFgCmBT29GKoLz6hKXGt+o3n3D0mjZGEFLdZRVYc5/H5pZwqZRrtfVR0kfN
A96j3FK/+2fMDZH7pswsU5vDbmKfpZYxrZiP/nR9yyYL/HX4XY9YLytjU1D7KUnLY/xp3+6n8Ab7yV+iZKNWw
pDkOOi9u9nLQ0NWdHVxtA06Fa/FwKSNgE9D5cGM01JIVWKfG8ZY0vlPc8+Js/L0n+ZHQcU4r7Uco3idDN6FWC
GoZljhRDjrqf+uAfqjrbP0WpJmozrLAdxCvdz4PhSX8Ad+Ib5Wg6c5krJlU8yOTD2S9Yb2ghN5BD7MRpIlKuX
QTHkJ/kDTVk6UuP7KPYnRwh8TaE60kZDtSW7Mv0sI8r7x4rnBs3rJbp5BZUcAYLXPuDH5+6oY7AGe4g7Oztg5n
5ifxqKeRHHEJKlinjh98WPhVBnA1jM9tjXnSSPTNTJ9bkUiUt4U7KpKZcCrsY5lWZAWh9vsgdlmqhSvNWqapTiC
iSb+XAUgA4I0MAxaIK1bmCQa0k95GefhQXLQDqgQMqNphCMdrlKBmX15lmXZu4P6N7+ruMbnEr9Z70IKBx7Bm0pp
PQrkJAAAAABSSchMAAAAAKCTlJmxAy5cvz6z6hRo/Liiy82dhQAANbCsmXLssMOO+Tpp59f59u7CgfWn5XBmB9UG7CBtS

8efNccsklWbFiRWNHAaCgli1bluOOOy5///vfGzsKbFRatGiRJUuWpKKiorGjfGj5XRmA9cHV0tfRnXfeucZzjzj
iiA2YhKbqyCOPzJFHHpljjz22saOssf/85z8577zzct9992Xu3Ll566236mx/9dVXGynZ+3vrrbfy3HPP1Zu7T58
+jZSKxrZ8+fLcf//9+dvf/pZBgwalsrIyL7/8ctq1a5fNNtussePRiN54441Vvle0a9a9eukdKsXocOHfLYY49lhx1
2aOwosFG56KKL8swzz+Saa65J8+bNGzvOh1IRf1eG9/LII4/k/vvvvr/fvkbFjxzZSKvhw8xN6HR155JFrNK+ioqL
JfiK5+eab1/uJdEVFRVq3bp0dd9wxxx57bI477rhGSPe2jh075i9/+Uu23HLL1eZdqakVbwcffHDOOeecPPnkk+n
Ro0c23XTTOtubYun91a9a9+NX/7299y/PHHp6qqqjArFh5++OEMGjQozz//fN79uU1T+jf4ox/9aI3nnnLKKRswyZq
78847c/DBB6dFixbv+6FOU/uafv7553PQPflhRdeyJIlS9K/f/9UVlZmzJgxJgxWbx4ca666qrGjpiBAwdm0qRJade
uXQYOHPiec2+77bYGSrV2Hnjj+//P4sXLLy6Pl0qlJvW94p0+//nP55e//GVOO+20xo7
ynor887o+ixYtyrJly+qMNcxYmw3nj3/8Y37/+9n7rvvvzm677bbK73Tn9XtykRTxd+Xk7Q9OW7dunccu3d
v7DhrrHPnzunXr1/69euXvn37ZqeddmrsSGusCN+TR48enf/+thjjbM89thjGzjNmvv
Rj36Ub37zm2nduvX7/j3VVP6Gou6GHuEoN9fRuz+JKaLzzjsvF1xwQQ4+wZk9uzZOfn
kk7N8+fKceOKJjZLsssssuS2VlZfm/m8oPhjVx8snJ6n/U7qm+kf0gw8+mAcffRf/OQnGzvvKWvnWt76cM7/
+9a+z9dZbN9mvk8uu6zO/VdeeSULFy5Mhw4dkiSvvZa2rZtm06dh8xHHnlamtr06lTp/f8UKpfkf2feuq
p2XPPPPfPnP/85W2yxRXn885OWRIOzomaunZD2fnP4GRbjLWbsMiJcvlY29mX/v3Qp/3XqCFp9Gt5PYvp5KDJG3ZNYXpLU9cnq
+wNlxxx3z/e9/P9OnT6/3j/+m8/v5TOvHO/y/12q3YqaEk5dLvQLj7PQ2d6tzHwcyFFJzPQ+U7XZz+3Cp5DW+MhY3kDpLRX+y2+eb
v+4d1U/pjukOHDvnCF7y/ti66Q29LLDf1ftOEi3SeJ0r4pl4xQ+ZRq84Ztp8ZnHKYGkZ3C9ZTpL9W8pnfZVV0+2MklpNlHLJzHr6rPz6N5iP8/XLfFqgHF5WFzOFyq
3b7ns3HnnnnRs7Yh1F+578wx/+MNdd12TXonjz9NEaGK3aCm9CBxhzsQ11PeHN+C9KBQ2nNaxJjqU8i0qPCkxTb31lXTXU7qwNaV
sX/nKV9K6detevV/p56p4qKiibzexENx2HpbFa1MS3gkSViiUUzJ3aA/NUbDFB0YlnP9zCmh6ss6thNAimFAjsx5HP92uuZJ7vYNjhp
/fnyuvvrqzJo1q5g33rSRs6f8+c/zUSFF3tC22muvrXOz9+fpcyW62zzAEdcy6L4UlZs2XHTNXd11BGmPx/6+g/ae1c7S2k96CIPjmc4eDf8XfyyFZBJfd2cR3wrLL0
Ew6N6Jp2HyE++//wxYvbhXnZxLvE4+nzJAMC+MAFT4Cl9kQq0+NOBlVQvsfWWw66W/a2KVcEgg3KSuwP7O4BtznbvvZP/Ujb8OTn6MSLE77mvo/6xsqq48q+a2uXc3Ij7E6
+GC/1H6vGyx6RV/wI9e6C6Ww3TM2a/IM5zyQL6MuUdRvqo9LY6aSGG3xkZNN/97+ax5XRqBVFgCkMhTXE5ZXzYpG2dRTg1dkRQ7SBTrNTN/bWDF7SS3PpiAVsUAvsSDYW/CO
t5LXW7dN2u4fC7Ul6xL9G8mv16e6t6lOqLQ7KN9wUR3JUkeNMOOK/zQ/vnwxfGv5nfdntMS5g/L3oR7nlS+L6wWnRTS8zuudz7nt/T89lrM3uC+JKevHT/P9cZnO96ntT7lG
yfZN6PmXaCldfk7BOw6C4+Gr+Lv3hnl2wWn2R1yvYnRK3B2c+54C3eSR3/1cr5l7e/c/ZkmWjHpvW/H/W+/51/3/5+nHrY3XXSz8PnRCKvFu+EF1MfvVRfR6nMVPwb6H
/3f8/t/jZQp6nhkvUjQMWTnCQ0EUZfbAfUIDBVsh7jf7X96flD9f8MdpXM7efqCyDLYfxLTmnbP/6hxRbnnTN+z/mxjOLXvWa/5RXNa8jWh27XfxhlrT+nDWaTGZ2vqMn77/
Dmy5fXlLuTPjnI+7I+rTr7vvcpGzP+qsDh+HULG96uR7XusVyjcX5+S8c03ruOnMz8fPzt7unPEwYuZaeWlxeTRR3xTCKw8xVjeOOUZUp/Rf8fpS4f7v9NlZzpzPn0fu/3
fu9P/fvnPCtTLoMtVFyxe3Xxrr1/nwwvUf/5Kx3NLvXdmQuh33Xsv7czbvlyWqppZ5WwfgJz7HEGqJD0JDgb+vrSXP3zTRnVbO/vXy3TZx98++3n/ZqyTLLG/vW+rTK/X+xr
9tvcg90x4TVxXT2+MKNDryptqq33+LFbj+fE0Tcz8e34xgr0OmfzTuRgjNE9Oo26f/H7EFDqG9e+fxEn5eVXvf2b/UQyyKZ8/zOnvvVMnYlcs4Fn8FPnKu/zUJvL3+8nFPX
hzrYa+rN3/rzsrzp4j/w2EpV48prMb9M/a2Ok4MDwDZPnQj4zHyOzOmCvnPq3ct20v8mrz+Pmu/rrjn6T4w5ZgFb/F5UyO++/l/ZLh+FvrnOQWYlsLHepP5hQmTjUcw1vjAD
oPLpaZW5Lr6xTtvS3V3WgNcWMnJKrM+ZXkd9bjqyPTG6d+SH/2fqeGAcWe/3V4wdPmPc7eSFkgoaXsK/JN+lYK/wx/v/HSzAcpjo1fW52DQxEzNb7/9vV/M/v/dXEiUWKQFjd
mWmGaxKjX8GjDPpN5N/vMDf9UI/rNfu/fnXv5fOPnHmrl39/75+o/BD5/QZXv7D7+Z6zrbSlYjLd/ntXCrWR3zZrOTfnHVveenJkOT5MyyhjzyvjZ21ZLz8uFn6Na+0XdMqhnze
XTv/i9PmyWH2v/eyhcX/rcvbjfs/vMVfG2dnfG+6trbAJFVN3mtvPOdcHnGDXLM2xlPG+DHnytkwdZ21XBN1erf+dTLd/YaZvZ8bJPZwSY0SCnUmqZ/DrLd6W96DeN7umy/Zd
0/S3i/LVedlr2OPP3RPHaXCAtZh9dqPKW5uOC+87W+Sv+pMrgAXb8RLrUQ9qQQV+ZhWR/zuflGeMK0RAlPqu4I/tVBvZAlfDgczN+2zGeNZ4jfNbHpY6JWVHRNbE8Tyv+WZ9Z
4qdJFIe8nLvrPU35L+f08fMOMEC/5n/ALdhMCGmaJpDZQDWlXW6JcZdcSHLD4B3/Q2GYVDN+69ix6L6IwxlLQHfnk6CRlJlfc+O1ry25zuGNhvxHZ/W7K3zJf+Hv0F2UD0TUQm
pkabVOh4rvhaYGN3mFKRO2K5o+S6pwjZJdSa4exY9NT7n1BFRO7Hwdn1TWqLCVO4hdZl35EQHwRiHZjYwnAdxFBVWwqbnxhMWe/TGfY4WHzV+Vx2V/Hd8g4+76GjjXjtsg
jwsqdwAFJnw6QaXVdCf72U94dI1b3jIrMyTlvHVyEtZP6zjeRjIWq5kvHGl3Pd7M4TR9Vvt7/Fjb2xcuTFYjWlXFYZpa3HGBGLv+ILMOGi6cKWQ0dvntHsdX7KXn/LbOWPIYZ
DjnHIwRCaJwFTlYJrCYWSibnJoF0MT9qScRIHb/zGSuIz1/NqcK2vXtcccOgJdnchN3bDw8Z9XM/u/AxE+bBhyzuWbgJAzp7NXbqvCd/XDUwyXO7p7/+DsNoZhPD9L3JuU+qOwE
mdoTGxsZ+ymw2QJnbbhtuNkGOA4fe82b/8CmkR+/eO4/6OS82NnL/IEH7zrKbQOuVx5zejtjg55Be5KxAh/BlLKq/p+8N+Pzv7otT9ZSDxWUiG/7ggvrqc6eyVszsCZL6eHoS2rBGb
3e8P2jVc0nkQsFrkTXnjIMOM9gxkBHR7s9a1mJdBr1jcyNJpfO4x6j9KZk2eyoYTWK37W9Ok7K+Wd9BTLlnV9dX3nVz5o7Kwmel4vNzkntz2S2u1Q5DusyTb/i7jA+87e9F2u7
bxDJVeRKTmB5uvOiyuzMxvz9rTnVWZfH4Wev0+yLrVFySiJ6/XWcm7FKLf5GZ+9GbfP1/vD2Tlrl1D7RrJNnK1HP/8zZlVDrJlmzm/nFuzBgXbZzzPCb3O0bBRVnUsrznSYyytqV
+n7u/WsnV/NWNuXObH+P4LTjn9Ulrj3NHtdd93nNurzYxRRp8bbv63Zn/bXH0rE9bqsdnGr69tQvntn5Ae2Q9ZEM4mTn0LV0bNEkurUnWgzKMiHY6DvvJfZ8Mjfvk+GnN5bNnJOaI7j3fnqMYz++4w25mQfcfPnF6+AyxZ
2rzjqH+ZzF0jK+9TqLjjSv8mQFT25iKXZ3Vv6V1HrUIUw6R5Z2IbpWwLJ1tQxqjtXmGq1txVdzcS3nMVqXXYR7VXeG3Q+jZXJyvwKp81rQvQqnK9GrKwxAdZncBeZO6+M5Tpp6FbdZ
HVLYUVrUEBKEpS/z+eKvtnF3Puvf8pzBM+5qr/B26Zn7bbv+Zgv2oWtNz9aOd1gsTfWvLf84QzbQYm1qYL1vvX8SunJFf5jNr0fEdNvrqNHzXqVyvxfJuuHtLR+lyLjhEDi9MHmm
nNHK+OjJ4/UJRXYmUNwT2oUlFZtodnsxxYp19zqOz3Cq8Tgy4lTjXqMXbp+ie+PzbbX9HkGBTsNt/mq5nq/nk3uFSEUb6/rdbV7sxKuekKfrHGs5yhVt2/vfvj0PxR1OcPtjv/e/
EScbP7Z6hsXh38vfSDtu8zdy2+7OD9ZHMnFuRYPZdrT1VudQoRdvPZsutUsm/U65fJ7fF9lTzkzIgHv3HPZS0oU4XiR3Pb7rLySzsR7PlRhkHvq9Ueo/GBxIOiobeaRdpObFA
Tiz7HjJVYeqdF23OxA2ku3cl7Wef4B3o0RnNxAWe0p9/8BVxNdhb7MBmlQrIy2YfrfKnnXYzbG+n3jXTGjZ5qpr/Oz8OOSq1fN3s8Kwz4dK/28WuZ6CcTwq+8Vmi9aU6OwH8+Ls
Lse2l4xW9g9TbVp9gs6W+xOTW0C94CyrasaQKc97s96gurRznFZHwZeNvs2eN+JB9hT+8G5ZxyrlBV/7Jbi28Hg0y2X8uT1IwuPP4/T1vn6kRC6Vr2Mv7eUE2qwKNPRKx3mIp5
3Y9Sz+8PhBHKhh7+9bFGx9QeGfrB5rnmaYc+TEfJTTvupA7eMucbVRTk3+yU1u/Lm/kKK3ru/6CWX9tsTzr65n5U9mjvXRoM1WnJ5svtXj8MuwaP1dy24ZlXiz7myNdXk9GP6Gp
5LhYfDuCadUyTzxJ1YlunvcdRT4qqUtU0GW5eE6i+x96uXVrHeiaMbuaZXXfvtY/+rvoDUjPbOWTCr+XqhcCY/UvjbYSQnmtxQUTL8Gzp+WB8eBGZG3dgM4mq9zOFd0d3Pd3
+6q0f+MnxK3mv8DDbSnx1nFNx9N05WJCPv6ahHbMrX+f9R8Hv9+H4PFNvCAO1xLb/Cns9JbDmyIF2IWnrGO1qSzJpivO2L4v3ar1Te5q2+fcPjvWxxqQ+ZNtjLs7sXOPPOqeWTK
dx/nv2v8C5U2u5m9lYhYllq0n1hLPS1nCLkh7UGzxGZnx4fdVMBSUy36pBM1tZxGHBIt87zOq5sWbO5VqP3e5vxoy29/Z/JXx2+X/47XPs+nvdWPXfVwUvXf5vT5Fx6BP0VNyHJ
y3nQaFT9s9cVzN3XVW8XPl3Y5uKb8u1HeMDOb8eZXBFuK3e8eGMMK2K99K1XtdMAf4PK8o9Bh5lPmCUzZO3M1/c9rUJunnDBvcff9PdnYQv3nCQCBZ2A9W4h/SJ/ZMVt3k7rMeZ
5z9+GaVPe9ekcKkEBJhuPcx0A0sxYqmdTLuk9/KcPf4jO/z3B4hl3+Fk5+AKVQH1YwK+Yq6WdGdNbDZzP/tFz+8nE1+nPQhdS7Kw3MpXvsN5eXr/bU6+x/ZkmONcZL+c3/97Uk/
LsXyf24znfi9MnuvQnaUXwVeGzvlOzPL8MlGmuWXrM/PnD3svSvdTdmf9Q4mdlDPH/3nzAPdYM77bk/X58d6LXuuL4MsR9T7mmc8MN9zTZN/w/OeQ0Tbm3V4nV3tdNmvuvNz6X3
mEeIYjhBG8mwoXlWnuJPL/S+YBdT/ZXMBJg9oWuCqWEpnZuXJcBXyrFLsOlI/VYH1+PdNtM/0HQDMm/zlhQyW8RNyTmYz6+fnFzcFuvfOtNO6nDBK2Hl/O7rXSDF8Ol9aHJ0
DZ8wmMJ8aIo+JKXeuT5Y2QNdAZFJN7SnlqSE09WKZJ5mAqqp2fRWQrl92anJ2m8x8gdw5o7yZX3cVQAfPDGeg1F9gHhcs0Vdvb3e0e5Y+c0D3YVmBMfOuIQq6uJ7/sWmsLffOFJO
I7T7MS1RfHJgz8W0Jcqh1/e9xyS8tSPp7O6Xj9BE/xw5oDXpPSU9N4NOCvP9SXVsH1EVVs8eHLZbtsxCDmMhqgKNyTxIa2/qmi/BJ12q6R2eKgsnWdEYIzKrKifLwxHB0x0B/3
cT4J4+DJWPTTqRUSGKr0sfbb6DXYhjnpJ4Tj5sfVP9+8O/jMaMRTsUgGPYN6z3Th09Rd4z2M5S6Sq3flu7y5w6g3mF9DL7rNvO6U+zQ9Yb3IeZ0S9rw8RDXebVv1Eq6PlZd2/e
rNsWfyRXiuL1yU6vJTmsGMpFzdnvlTTf+r4rXqR68EGXpvuLucZZprQRjmYdzkVsZ0MzjvrhlGt1mUwJS34D4tmmSz0IQJsC9lMO7PnvmBtJtDXgjdBwX95WyUrbMyOtZLu1zfPrC
0tjyWrunfd/UI6hRt27hXpWkG/Yrbx/uXXkNs2Iq4HNqsN1PhfXyZjh2I5eYU+bpkfjB7l92Q1OIF0cgyT3dwvvMOPo9S3nR83v7XJBSOwCSJZcLqJXqo0vlFW0SxyQtkjx6RfNo
OhcSHyYVvCj8y/S+9T9+rrrmu+a3FMM0RU3B6bQNrDtwxXpAN3/eTmdTI5VcT6fSd9O8ruBYM5BeYg2LHLbAqXlbtaRG04n5lqWQnJX5IezPPz2ebcBhYKL4+yLpBtvOP2oqMHZ
RvHh28bS0ZZJhsBhnH1k7bdJP/fh4bBsLfH4yKO3YW3JYJd3v5GyOqs9XLk/TbH8MKLfeAKm1e3Uj68rmYIVoRm0h3f+HQUq0tzz/KkJvNAj78C33OgJ88ipTjWmGjtS5ygJ0xRvv
xfPWQLP6W9NNJh8dxffn6lxIHhnIl3f1IVYZqXKr8PQjsv2Sc1S4+L1ZXpbdQ6Yzwv86h18dFYYM7MJxTfvXVzbZG4u0dNN+2L6Lp1Yln2FnwW7SODHNUkSvMnT7++eQD1MeGkS
LrObnm7Tz8vAbrwZ/e/Z7tnT9vjo3ahm9OpSWRMzOYcyoVoWTGSxNoewtuGWVGtOFStzxHbJnCmJnO5brndxn3KYgNuN+Jsr/8p3k5kGrvHObBTI4nkbc8DwSVqPLd7n+/OPMsi
Z8UhkzKdL1U1QTmFEVyPkePZ9zT2bPcEVYiWnKKdXeIKqRD4xZH4p8GpUGk+0sgcTTG12eqm7dRvfn+LVnWcsMq5nR3DCmjWbzD92zPWdnwbgVCNWS9p0vLG69aVy8zZqVe0aDn4f
ekyJTZNQm4PKhw0nEDG2r5DnjA7jDIeWDodjr8ffvr2k8lV+V3cRLOd98NtvDHA6DNqd8GEl7sOzVWzUJ+rB9pfqjXkNYprMr9zxIrafqKEpE7Ia5N4AdBgYp90kB05/V8cL9TTp
O7fVbUnW/9kRDxgO2gGyxDb7OzNe9c+Q2ePDm82D/YOxBrzxy+35HWqtG1mU08bZUb28J4zOHRUqL6xF7qH+03WoXs70hGDTUvUvJ97fgNUXxJ6Qov2H1JCxTtjS2EzrxHu+eUvX
8efNccsklWbFiRWNHAaCgli1bluOOOy5///vfGzsKbFRatGiRJUuWpKKiorGjfGj5XRmA9cHV0tfRnXfeucZzjzj
iiA2YhKbqyCOPzJFHHpljjz22saOssf/85z8577zzct9992Xu3Ll566236mx/9dVXGynZ+3vrrbfy3HPP1Zu7T58
+jZSKxrZ8+fLcf//9+dvf/pZBgwalsrIyL7/8ctq1a5fNNtussePRiN54441Vvle0a9eukdKsXocOHfLYY49lhx1
2aOwosFG56KKL8swzz+Saa65J8+bNGzvOh1IRf1eG9/LII4/k/vvvr/fvkbFjxzZSKvhw8xN6HR155JFrNK+ioqL
JfiK5+eab1/uJdEVFRVq3bp0dd9wxxx57bI477rhGSPe2jh075i9/+Uu23HLL1eZdqakVbwcffHDOOeecPPnkk+n
Ro0c23XTTOtubYun91a9+NX/7299y/PHHp6qqqjArFh5++OEMGjQozz//fN79uU1T+jf4ox/9aI3nnnLKKRswyZq
78847c/DBB6dFixbv+6FOU/uafv7553PQPflhRdeyJIlS9K/f/9UVlZmzJgxJgxWbx4ca666qrGjpiBAwdm0qRJade
uXQYOHPiec2+77bYGSrV2Hnjj+//P4sXLLy6Pl0qlJvW94p0+//nP55e//GVOO+20xo7
ynor887o+ixYtyrJly+qMNcxYmw3nj3/8Y37/+9n7rvvvzm677bbK73Tn9XtykRTxd+Xk7Q9OW7dunccu3d
v7DhrrHPnzunXr1/69euXvn37ZqeddmrsSGusCN+TR48enf/+thjjbM89thjGzjNmvv
Rj36Ub37zm2nduvX7/j3VVP6Gou6GHuEoN9fRuz+JKaLzzjsvF1xwQQ4+wZk9uzZOfn
kk7N8+fKceOKJjZLsssssuS2VlZfm/m8oPhjVx8snJ6n/U7qm+kf0gw8+mAcffRf/OQnGzvvKWvnWt76cM7/
+9a+z9dZbN9mvk8uu6zO/VdeeSULFy5Mhw4dkiSvvZa2rZtm06dh8xHHnlamtr06lTp/f8UKpfkf2feuq
p2XPPPfPnP/85W2yxRXn885OWRIOzomaunZD2fnP4GRbjLWbsMiJcvlY29mX/v3Qp/3XqCFp9Gt5PYvp5KDJG3ZNYXpLU9cnq
+wNlxxx3z/e9/P9OnT6/3j/+m8/v5TOvHO/y/12q3YqaEk5dLvQLj7PQ2d6tzHwcyFFJzPQ+U7XZz+3Cp5DW+MhY3kDpLRX+y2+eb
v+4d1U/pjukOHDvnCF7y/ti66Q29LLDf1ftOEi3SeJ0r4pl4xQ+ZRq84Ztp8ZnHKYGkZ3C9ZTpL9W8pnfZVV0+2MklpNlHLJzHr6rPz6N5iP8/XLfFqgHF5WFzOFyq
3b7ns3HnnnnRs7Yh1F+578wx/+MNdd12TXonjz9NEaGK3aCm9CBxhzsQ11PeHN+C9KBQ2nNaxJjqU8i0qPCkxTb31lXTXU7qwNaV
sX/nKV9K6detevV/p56p4qKiibzexENx2HpbFa1MS3gkSViiUUzJ3aA/NUbDFB0YlnP9zCmh6ss6thNAimFAjsx5HP92uuZJ7vYNjhp
/fnyuvvrqzJo1q5g33rSRs6f8+c/zUSFF3tC22muvrXOz9+fpcyW62zzAEdcy6L4UlZs2XHTNXd11BGmPx/6+g/ae1c7S2k96CIPjmc4eDf8XfyyFZBJfd2cR3wrLL0
Ew6N6Jp2HyE++//wxYvbhXnZxLvE4+nzJAMC+MAFT4Cl9kQq0+NOBlVQvsfWWw66W/a2KVcEgg3KSuwP7O4BtznbvvZP/Ujb8OTn6MSLE77mvo/6xsqq48q+a2uXc3Ij7E6
+GC/1H6vGyx6RV/wI9e6C6Ww3TM2a/IM5zyQL6MuUdRvqo9LY6aSGG3xkZNN/97+ax5XRqBVFgCkMhTXE5ZXzYpG2dRTg1dkRQ7SBTrNTN/bWDF7SS3PpiAVsUAvsSDYW/CO
t5LXW7dN2u4fC7Ul6xL9G8mv16e6t6lOqLQ7KN9wUR3JUkeNMOOK/zQ/vnwxfGv5nfdntMS5g/L3oR7nlS+L6wWnRTS8zuudz7nt/T89lrM3uC+JKevHT/P9cZnO96ntT7lG
yfZN6PmXaCldfk7BOw6C4+Gr+Lv3hnl2wWn2R1yvYnRK3B2c+54C3eSR3/1cr5l7e/c/ZkmWjHpvW/H/W+/51/3/5+nHrY3XXSz8PnRCKvFu+EF1MfvVRfR6nMVPwb6H
/3f8/t/jZQp6nhkvUjQMWTnCQ0EUZfbAfUIDBVsh7jf7X96flD9f8MdpXM7efqCyDLYfxLTmnbP/6hxRbnnTN+z/mxjOLXvWa/5RXNa8jWh27XfxhlrT+nDWaTGZ2vqMn77/
Dmy5fXlLuTPjnI+7I+rTr7vvcpGzP+qsDh+HULG96uR7XusVyjcX5+S8c03ruOnMz8fPzt7unPEwYuZaeWlxeTRR3xTCKw8xVjeOOUZUp/Rf8fpS4f7v9NlZzpzPn0fu/3
fu9P/fvnPCtTLoMtVFyxe3Xxrr1/nwwvUf/5Kx3NLvXdmQuh33Xsv7czbvlyWqppZ5WwfgJz7HEGqJD0JDgb+vrSXP3zTRnVbO/vXy3TZx98++3n/ZqyTLLG/vW+rTK/X+xr
9tvcg90x4TVxXT2+MKNDryptqq33+LFbj+fE0Tcz8e34xgr0OmfzTuRgjNE9Oo26f/H7EFDqG9e+fxEn5eVXvf2b/UQyyKZ8/zOnvvVMnYlcs4Fn8FPnKu/zUJvL3+8nFPX
hzrYa+rN3/rzsrzp4j/w2EpV48prMb9M/a2Ok4MDwDZPnQj4zHyOzOmCvnPq3ct20v8mrz+Pmu/rrjn6T4w5ZgFb/F5UyO++/l/ZLh+FvrnOQWYlsLHepP5hQmTjUcw1vjAD
oPLpaZW5Lr6xTtvS3V3WgNcWMnJKrM+ZXkd9bjqyPTG6d+SH/2fqeGAcWe/3V4wdPmPc7eSFkgoaXsK/JN+lYK/wx/v/HSzAcpjo1fW52DQxEzNb7/9vV/M/v/dXEiUWKQFjd
mWmGaxKjX8GjDPpN5N/vMDf9UI/rNfu/fnXv5fOPnHmrl39/75+o/BD5/QZXv7D7+Z6zrbSlYjLd/ntXCrWR3zZrOTfnHVveenJkOT5MyyhjzyvjZ21ZLz8uFn6Na+0XdMqhnze
XTv/i9PmyWH2v/eyhcX/rcvbjfs/vMVfG2dnfG+6trbAJFVN3mtvPOdcHnGDXLM2xlPG+DHnytkwdZ21XBN1erf+dTLd/YaZvZ8bJPZwSY0SCnUmqZ/DrLd6W96DeN7umy/Zd
0/S3i/LVedlr2OPP3RPHaXCAtZh9dqPKW5uOC+87W+Sv+pMrgAXb8RLrUQ9qQQV+ZhWR/zuflGeMK0RAlPqu4I/tVBvZAlfDgczN+2zGeNZ4jfNbHpY6JWVHRNbE8Tyv+WZ9Z
4qdJFIe8nLvrPU35L+f08fMOMEC/5n/ALdhMCGmaJpDZQDWlXW6JcZdcSHLD4B3/Q2GYVDN+69ix6L6IwxlLQHfnk6CRlJlfc+O1ry25zuGNhvxHZ/W7K3zJf+Hv0F2UD0TUQm
pkabVOh4rvhaYGN3mFKRO2K5o+S6pwjZJdSa4exY9NT7n1BFRO7Hwdn1TWqLCVO4hdZl35EQHwRiHZjYwnAdxFBVWwqbnxhMWe/TGfY4WHzV+Vx2V/Hd8g4+76GjjXjtsg
jwsqdwAFJnw6QaXVdCf72U94dI1b3jIrMyTlvHVyEtZP6zjeRjIWq5kvHGl3Pd7M4TR9Vvt7/Fjb2xcuTFYjWlXFYZpa3HGBGLv+ILMOGi6cKWQ0dvntHsdX7KXn/LbOWPIYZ
DjnHIwRCaJwFTlYJrCYWSibnJoF0MT9qScRIHb/zGSuIz1/NqcK2vXtcccOgJdnchN3bDw8Z9XM/u/AxE+bBhyzuWbgJAzp7NXbqvCd/XDUwyXO7p7/+DsNoZhPD9L3JuU+qOwE
mdoTGxsZ+ymw2QJnbbhtuNkGOA4fe82b/8CmkR+/eO4/6OS82NnL/IEH7zrKbQOuVx5zejtjg55Be5KxAh/BlLKq/p+8N+Pzv7otT9ZSDxWUiG/7ggvrqc6eyVszsCZL6eHoS2rBGb
3e8P2jVc0nkQsFrkTXnjIMOM9gxkBHR7s9a1mJdBr1jcyNJpfO4x6j9KZk2eyoYTWK37W9Ok7K+Wd9BTLlnV9dX3nVz5o7Kwmel4vNzkntz2S2u1Q5DusyTb/i7jA+87e9F2u7
bxDJVeRKTmB5uvOiyuzMxvz9rTnVWZfH4Wev0+yLrVFySiJ6/XWcm7FKLf5GZ+9GbfP1/vD2Tlrl1D7RrJNnK1HP/8zZlVDrJlmzm/nFuzBgXbZzzPCb3O0bBRVnUsrznSYyytqV
+n7u/WsnV/NWNuXObH+P4LTjn9Ulrj3NHtdd93nNurzYxRRp8bbv63Zn/bXH0rE9bqsdnGr69tQvntn5Ae2Q9ZEM4mTn0LV0bNEkurUnWgzKMiHY6DvvJfZ8Mjfvk+GnN5bNnJOaI7j3fnqMYz++4w25mQfcfPnF6+AyxZ
2rzjqH+ZzF0jK+9TqLjjSv8mQFT25iKXZ3Vv6V1HrUIUw6R5Z2IbpWwLJ1tQxqjtXmGq1txVdzcS3nMVqXXYR7VXeG3Q+jZXJyvwKp81rQvQqnK9GrKwxAdZncBeZO6+M5Tpp6FbdZ
HVLYUVrUEBKEpS/z+eKvtnF3Puvf8pzBM+5qr/B26Zn7bbv+Zgv2oWtNz9aOd1gsTfWvLf84QzbQYm1qYL1vvX8SunJFf5jNr0fEdNvrqNHzXqVyvxfJuuHtLR+lyLjhEDi9MHmm
nNHK+OjJ4/UJRXYmUNwT2oUlFZtodnsxxYp19zqOz3Cq8Tgy4lTjXqMXbp+ie+PzbbX9HkGBTsNt/mq5nq/nk3uFSEUb6/rdbV7sxKuekKfrHGs5yhVt2/vfvj0PxR1OcPtjv/e/
EScbP7Z6hsXh38vfSDtu8zdy2+7OD9ZHMnFuRYPZdrT1VudQoRdvPZsutUsm/U65fJ7fF9lTzkzIgHv3HPZS0oU4XiR3Pb7rLySzsR7PlRhkHvq9Ueo/GBxIOiobeaRdpObFA
Tiz7HjJVYeqdF23OxA2ku3cl7Wef4B3o0RnNxAWe0p9/8BVxNdhb7MBmlQrIy2YfrfKnnXYzbG+n3jXTGjZ5qpr/Oz8OOSq1fN3s8Kwz4dK/28WuZ6CcTwq+8Vmi9aU6OwH8+Ls
Lse2l4xW9g9TbVp9gs6W+xOTW0C94CyrasaQKc97s96gurRznFZHwZeNvs2eN+JB9hT+8G5ZxyrlBV/7Jbi28Hg0y2X8uT1IwuPP4/T1vn6kRC6Vr2Mv7eUE2qwKNPRKx3mIp5
3Y9Sz+8PhBHKhh7+9bFGx9QeGfrB5rnmaYc+TEfJTTvupA7eMucbVRTk3+yU1u/Lm/kKK3ru/6CWX9tsTzr65n5U9mjvXRoM1WnJ5svtXj8MuwaP1dy24ZlXiz7myNdXk9GP6Gp
5LhYfDuCadUyTzxJ1YlunvcdRT4qqUtU0GW5eE6i+x96uXVrHeiaMbuaZXXfvtY/+rvoDUjPbOWTCr+XqhcCY/UvjbYSQnmtxQUTL8Gzp+WB8eBGZG3dgM4mq9zOFd0d3Pd3
+6q0f+MnxK3mv8DDbSnx1nFNx9N05WJCPv6ahHbMrX+f9R8Hv9+H4PFNvCAO1xLb/Cns9JbDmyIF2IWnrGO1qSzJpivO2L4v3ar1Te5q2+fcPjvWxxqQ+ZNtjLs7sXOPPOqeWTK

2RWBgIG9XYZG6R8XNj6h58+YoKSmBg4MD7Ozs0Lt3b5ibm7OOVe/Y29sjJCQEWlparKPUirKyMjIyMqCvry9VqEh
LS4ONjQ1ev37NOqKM3r17Y8mSJbydOShPSUkJTExMcOrUKZiZmbGO88Fu3bqFjIwMcBwHU1NTtGvXjnWkamlqauL
PP/+EiYkJNDU1cfnyZZiamuLPP//EhAkTkJGRwTqihBA/FyoqKkhPT4e+vj50dXURGhqKTp064fbt2+jevbvcGTh
8UHWgQSwWQ0dHh5dLmoS4+ZiNjQ2cnJzg5+cn+S7R1dWFm5sbnJyceLk0T2jf19XJzs5GQkICjIyMeFl0s7GxgaK
iIry8vOQW61mvvJFHVVUViYmJvBr0+FCZmZno3LkzXr58yTqKXELqUyik+4qqysrK4O/vX20rGT7tSl+dxMREbNq
0Cfv27ePlhkJV8f2arK+vj3Pnzgn6+iYELVq0QFRUFK+fmUjdomXpH1Hz5s2Rnp6O7OxsZGdn48GDBzAwMJBajsV
X5eXluHPnjtx+Fb1792aUSr7IyEjWET5Ihw4dcPHiRZmZQIcPH5bZRIYvZs2aBS8vL8ybNN08wvSuVlJTw9u1bwS5
JaNeunWC+nJWUlCS/52bNmiE7OxumpqbQ0NBAdnY243TShPi5aN68OfLy8qCvrw99fX3ExcWhU6dOuHfvXrU9DFk
rKSmBu7s7tm3bJojPsYODg+CKm+np6Zj3A0IEn6vCoqKiiguLkbjxo3h5+cHV1dXXhY3hfZ9XSEwMBCjRo1Cw4YNAbwrErV
u3Rp///03AgMDMX78eMYJpV2/fl0wqy0qamqmJmZ4dmzZ6xj/CtHjzbz/CtHjzh9e7YQupTKKT7iqp8fX2xY8cOfPPNN1i8eDG
+++47ZGVl4ffff+ftkt6kpCRERUhKioKFy9exKtXr9CpUyd4eXnB3t6edbz3qrgm9Nq1CyoqKiy9ewKnyXKnB3t6edbz3qrgm89WyZcuwdOlS7Nq1CyoqKqz
j1Fs+Pj5Yv349Nm3aJKj7fPLpUHHzI7p27RoRoKCgpw4cIFREdHY/Hixbhx4wwYsLCxgb2+PVatWsY4oV1xcHMaOHY
79+/LPDjzdTmTUPqOAe968YwbNw5//dNhNNAH4ihHpStWMBCnTp1iHU+uUaNARBW70rgXVH2p59+wo4dO
oKJzLm5A+z8C7nUETEhLQrl072NvbjZ7SZ22NI0eOIC2TLzZxkleNJPdhTRfNGWwAIyXQmUb3HJdlwbwNFgxpuSYJqu
L85SUlHD9+nXB3FwOGTIE3t7eSE1t7eSE1NlTuA4+LiwihZ9VRVfH27VsA72Yr3L17Fx06AwAPw84OTn
JtAV59eoVPDw8eFfctLa2Rk5ODu+Lm5000+YP38+Vq5cKfdvk+E99Qi0tLaWubRzHITc3F0+fPsWHl7zfPFs3F0+fPsWWLVsYJqu
ZkPoUCu2+orL9+/+/fjt99gYGRnt7ZKlWqQ3D1B1NrvDCmn2cXbxnoqLjzaarNzCvVoNqgICRVAZzmB4vs5VzkqPKrpLKzGcYKbWnPymY0m2o/OXnCw8O
r3TgmICCAUSr5nNmzYgt376JZs2ZnpJeVjgo5IPNzAPPze7+s3JYyLu62Gncw9/FVX6jtvQ0ZaTf1ssbBNNdeMsRjk4jmb3b7xrxZzm4ZsCrrPHMbz2rUiQtV
jx4/jwIEDvJ7i37lzZ7R7Rr1w6+vr5y+SW6x5zrjUvTwhRKZr6Ac9erF9peHSWUDewAbs9ErbEuAbBzGvS8WqAc7FBmkVNTrkqXVZ1fQlu5KEtNPS1fYPgzJo
sGMLDw9G4cWOYm5tDVVVV6jwfv+SE+HlOSEjAq1evYG9vLcbZvTzkbGhMT0ANJY+zatYt3S4SE9rkSoLy9sSeX5mZbAFhbHBw
s+f1OmzaNt7tM+/j4QElJibcDeZWJxeJ6Jxqz/F1aGAfo0KFdwdbGZmMTMX/+eBw7dfgzu7u6SZd9hYWGsI8oQ4vUNePf
5ePz4MXR0dKSOJycnw97Jycnw97enlcPesC7lSDLli3j/WqLir68FSoGTCvj4y8FqrP6+v1OuKlht2dna8XnYqpD6FQruvqEx
VVRXp6elo3bo19PT0cPr0aVhzWEzMxOWlpZZ48eF64gyxNr58yftiZmWW+vr7w8/ODtbW13GfVYb8eOMUomX9VrrlV
824RqyZIlNc4+5lOBvvJGkO/D137I5NOh4uZHdOzYMckU/xs3bkBbWxu9evevCnZ0d7O3tJTMs+EzovceE2HeM1I3
3feHx8UuOPs+fnhA/F0I0a9YsBAYGwtjYGNbW1W1jjJFZL7O0hOKzMxMvMvH79GhYWFigqKsLcuXMD//v68HHQS2vW
tYoZecnIyOnToIDXTu6ysDPDPfu3YOTkxOCg4MZppppQlr1jPx9UW0dHRtX4vH/uECt39+eRmJjI2z6FQeiAwEB
069YNvXr1grOzMxYuXMjxehnVp6p6hbDtPbJZoR11g2IMRrUlb2qbOnj0/...

xcXGciYkJy2jVUlVV5e7du8dxHMcNGTKEW7VqFcdxHHf//n2uUaNGDJPVP4sXL+b09PS4n3/+mWvUqBG3fPlybuL
EiZy2tja3fv161vFkDB48mHN1deWePHnCNW7cmEtLS+MuXrzI2djYcBcuXGAdT65WrVpxx48flzn++++/8+57pFO
nTlx4eLjM8fDwcK5jx44MEtVO48aNudu3b8scv3XrFqeqsogUfVMTU25q1evso7xj4hEIk4sFnMikUjqX8UxsVj
M9e7dm8vPz2cdVcLFxYXr3bs399dff0mOPXjwgOvTpw83dOhQhsnqBxUVFe7+/fscx3Fc8+bNucTERI7jOO7u3bu
curo6y2j1zvPnz7nffvuNW7hwIZeXl8dxHMclJiZyDx48YJyMsEDzY+oVatWkgdoNTU1yQ1FYGAgN3DgQIbJ3u/
OnTvczJkzzub59+3L9+vXjZs2axd25c4c4d1LLnc3d25sLAwrry8nHWUAr148ULUyLzw8nPv888888+5yMhI7tmzZ1LnXrx
4wTqqXD179uSOHTvGcRzBQYGcnv37uUSEhJYx6nRtm3buH79+nEPHz6nNGE+ZSVlSXFq8quX7/OqaioMEh
Usx07dnCtW7fmli5doQ7fvy41D+hWbp0Kefj48M6hpSmTZN2/e5DiO049q1a8edPXuW4ziOS09P5+0aOzivS09P5+0Ajo2
NDbdgwQLuwoULXKNGjSTXt8uXL/OuGFTV27dvuZycHO7+/ftS//ftS//jK0NCQO3XqFMdx7wpEFfdD69ev58aMGcMymlz
a2tpccnIyx3Ecp66uLinch4eHc507d2YZrVrz5s5s3j9PX1uYiICK60tJQrLS3lwsPDOX19fd5dL06fPs116NCBO3z
4MJeTk8Pl5OzNzbnTp0/z9nzzNWxDi5Ijw7O79tQjOqWxkPw/W5RQtGZoNf58zr1xWMc9Fvzr3x09gAJNaH/PPLCQvMAKDmnW0BtJhcXNtjfGgO79Lt0CfVYvGcOJvnBOwYORvUYtTEpHjlb9PPHCNAlbkYSJP8XvJUk+mVxvJMkDmL/RjS/vjYfMcUJ9hA6TYyyNsTbstywqGVVjHuNYc9H37rhHNLyxnZk/A1b9w8x8lQYbvV9jMiEPvMEHBNcqDS45Mo7/KNKX1vXIy8xXC6PiGEnBuxKpZ/F0AbUZLu8t9HMbbfiNx3TY8AI8ovjN1JGUVSfjaJ1DYhE8SUu7nYOtDTY/lxZuiEklJzDjfNtOE0Gi6Jt7FDmHs6jpSi4ftbjRhJvHQEYj6dWtL+Yl5amtzVl7B9U4cKl6d6T4aV4ajl6yZKPXXs1jcvaT4GbXfaSUG/LuD/eHzhQnIDomrYWbGqJzEgE9fGbzBjdkcgRGtrVKGp0sapzTW7fPGtT/J2g/Pmj+Vjb5i60dJVQNEKjCkUh06L6BOs7/pZTBLwCvOs6wIrTJ/m5FvQ0wPnrr0i3eKtQU1r1V/Tq6U4t8jYNOj9UIVd66i7FBEoYWhk5JrGs2zG7DHG1AsCQmm1QP+3JuYlKaPKyvaDJkSV1U3zMGg0xv0XOydAc8uAEXydX6PePBUhvwAU+AVtfQlZLHa/MU1ZkDeKRGXRWdlFqxSsd4s6i/Pfo0Y9ZvWm9n6jQjqWaYxhqwwtTGuTbpfWQt8FKdqdj5AfMJA7TQaWLzf4VKm0crIaDkOeYSOXbF3D0/pPIzHu3tGL7Zl7z4Kdt09fs2jOHA6rW4+qVAkyXtEqjOkEHN4h+H4+HQZQaLqnoBu/t/mUVpTPe2x8sWUOhfOgzfK8rjOAkR1cfaIoIiWsX/mDbTCVEMTYnCpeDiXGtr1eXvBcJyDNpJ3vDz6DwVdIAeB22e+kKELHTlHX

EiRMwMjKSHH/8+DFatGjB25sfjuOwbt06/PLLL3j48CEAoEWLFpg3bx5mz57Nu9G8li1bIi8vDwMGDICbmxuGDBn
Cu35ulTVq1AgZGRkyvUCzsrJgamrKy2XHlpaWuHPnDkpKSqCvry/z5XD16lVGyWr2vuUrSkpKGDVqFLZt28b0M1N
1Nk1hYSFKS0sly+krflZVVUV+fj6rmNXq0aMHAMDLywvNmjWT+Z336dOHRaxqicViPH78WLJcrMKtW7dgbW0tVYz
hA6E9dAjRwIEDkZ2djZkzZ8otILu6ujJKJqvydU3eLZuysjI2btwIT0/Puo4mV9XrcMWgR2V8HQipoKenh9jYWJk
lhUePHpUsNWXtQ/pIh4SEfMIk/4yBgQF8fX0xfvx4qeN79uzBsmXLcO/ePUbJ3rl//z5at24NkUiE+/fv1/hePvY
JBfhfQK7s3r176NmzJ+bOnQtvb2+8evUKAwYMgKKiIv744w8qUnxk2traiI+Pl3peFYLrGLg2+zpFSt
WYM2aNejbty+6dOki8l0eO2qtoWivn8XV0hMjISiYmJKC8vh5WVFfr168c6EmGElqV/ZKamprCxsUFwcDD69u0
rOc7nGrJIJIK3t7fkZgIA1NTUGKeq3pIlS/Dll9KbajPPcQ6qyUY8eOYcGCBZg
3bx5sbGzAcRyuXLmCX375BUuXLkVpaSkWLlyI77//HmvWrGGWU+hN9FNSUpCYmAImvKh7+RSIR3N3dpWbtlpW
VISUlRVKo5ZO///6bl7nk+ZBlr3wq1MfExODPXAcB0NDQ8THx0sV6eRs0aAANDQ8THx0sV6Rs0aABdXV0oKCgwTCh
NqJuvVDZ9+nT07dsXly5dgp6eHgDg0KFD8PT0xO7du9mG+39V20AIza53r06IFHjx4xSCStcsGSSr8XL99m
4cSN+/fVXqQKyq6srOnTogGXLlvGquGYBaNiwImZhYSGio6ZhYSGio6PlFt74UMiqatatJKSqqoAe5Pb8fcXFxj
kSThw4IBglk4/ffoUHh4e+OOPP+Se51vxbceOHdDU1ERiYiZI1ExOlzrHawKsqoW9eWV5ejt27dyMkJARZWVkQiU
wMDBA8+bN5Q6qkv8GKm5+RCKFu2bMH+/fvh7OyMDW4MTpEKKPA/SNc2mUnv/WnNU0LMaFfOmaTSpfvuYGtdPdy
dHQ0vLy+MHj2acr5hLrxx4oVK7B/wcwjBEMo+XBdADvlwHQqTxp8PQqC/WtWrXi9aBjZRWFFaE8iPBt5vY/sWTJEEuzPl4SSp4/R4xyZIGc3+aw
9i0mTJmHv3r286eO1a9cu1hFh+HbHyW9LGgdRA4uV49nqttDvsLYrhh/0GZnAaAs/LuCmkJrUqM02g2ZY8Cvk8cTuAKPGqqksLhz4dpzNvfVY/
QrVs3nDp1itcbCVFIIsKrCoEGDUFRUhMLCQjRp0gTPnj2DioqKkJYMcWxCVCYT5i9NT+ITTusd2fkzYDHFz3xNkSIRZVXEVLJwy5FqM8M9FI
SyTdnzhw8f/4ccXFxsLe3x7x5fLmXFnxSU69dsqhmwcBc1LGJyI07w9JArKEuDj4sbcNMqMYt5cFE3F
HSEgIfv/9d9tQXPCvnNxWW2G1WK4yqYYSMmoXOWDsnD0CMkAUPr3M4Ky3TT7NzN/nymmxmZuZcg7/2Uf
+wP3799G3b1/JsPzn3Ff1b36Q8f8V8a8ypJmGq8dYbL6oZpqJeu7qUkmUdnz0dMZNU3j1LJDNwALu9/M0YZQkFDz
111/o2LGjzE0xX5YTz8t2nTbnzhXE7A9AWA8dQi3Ur1u3DgsXLsS2bdtkZtXzyYKTJzBw4EAoKSlJ9ceh68
FFkA4SwkrW79+PcaNG4fu3bvjr/+wsGDB3nVrkDofH19MWzUKFy4cAG19MWrUKFy4cAG2trsSfu9hYWE4fPgw63fSM
MjMzMWZYMKS
mpkr13qy4j+bbrLEKNRWQq24iykJ1zyINGzbEw4cPYWtrKznGx2eRCt7e3hgyZAh+/fVXaGpqIi4uDkpKkvjjqq6/
g5eXFOp5cKSkpkpUL169fZxumFUmTsZNHnHHVTVcsE4v5Cho28Zmxx+j2U0m8s8AeJM5vfE8x9Vi8ZFs53rNtjmDBiUdlzbA3fh7qkYz/LUENX/NeCKs
IDw+X2dMiIiCQ4cORWBgoEzbE1L/Uc/Nj6hbt24QiUQ13hzMnz+/DhN9mZzUKDRbbFb7PjRVWYfvrqWgDm9TgsWrvQuNvGQ3Qivi4sLvfRUcFITSHFz48ZPRDdE
RoaGhcHR0xO3bt5Gbm4jw4bydGwDrVm4thw4bxdjbDrVu3kJycDGVlZ/Gibm/N6mdP7elfy6bNcmaWlTJp16oTt27ejjQ0YMGAICSkhJ
MnjwZycnJSEpKQmxsL766ivejKoWFhZ9iwYIFCA4ORl5ensx5vm5ECQmxVWl4/WsycmsLoQvAdG80ko22PSJl1NACoHxosL1YkILHQIaKzG4RsIsE65AG9OdDfbvFlUN4Ixbr+yfrQqsmGPbmvkrk1/PaKaJQTf9eRP9xlRZjXzbY3AGAk/qbxJD5OBzbMqXgJM8L/3zaRm/SJqBAdeIZINqLqvZ/Sx4WO+FZINQx4ZLY0+cNFGDWKHKSpKKVbh1ClN9jZNQAMl+4aqtLQtyGebbkJ+xZFvZdZe+J3BQ/v1b2hcA
REVGHaf4ZvhfqtbS0UFRUhNLSUqioqMgUkPmyhL7qpibV4evfnZCWEsorHpeUlMDb2xuOjo5Sx+0WNdDuVwoqKnP1RE1DfBC5h2NahA7ONDpUSkj8IGHJ+VS
...

4f/48NmzYgKysLGRkZKB169Zo2rQp65gyfv75Z4SGhko9+6urq2PZsmVwdHSEl5cXlixZAkdHR4Yp/0dos4/Lyso
k+1jIo6CggNLS0jpMRPiCipv/kpaWluQBSVNTU+6DBx/70Al1h1sA6NWrF0JDQ2Fubo6RI0fCy8sLERERCA0NrXa
JRV0T6rLSmhQXF2Pjxo1o2bIl6yhShL7hhqGhIbKysqCvrw8zMzMEBwfDxsYGJ0+ehKamJut4cg0ZMgTe3t5ITU2
Fubm5TL9Cvv2es7OzJTvGKisrSx5CKjYL2bRpE8t4Mry9vaGkpCS5uawwatQoeHt787a4KaRCfXWLVt6+fSvVXoZ
Pxo8fj507d/K2XYU8QlpK+CFL2Pi2uUJ1jhw5wpuebpVNmjQJ+fn5GDp0KI4fP47FixcjNzcX0dHRaNGiBet4Mr7
//nsUFhYCeFe8GjJkCHr16gVtbW0EBQUxTle9iv7Y9+/fl7nm8e25ZNasWXj58iVu3Lgh+d5LS0vDhAkTMHv2bBw
8eJBxwupZW1tLftbR0eHlipuqPDw8kJeXB0dHR8TExODQoUNYuXIl/vjjD8n9Eh8sXboUu3fvhpubG5SVlXHgwAF
Mnz4dhw8fhpWVFet41Xrx4gWePHkis+T86dOnkslQmpqaMm2SWLG0tJSsrLC3t8eSJUvw7Nkz7N27F+bm5qzjyeA
4Du7u7mjYsKHc8zRj87+Lipv/UkREhOTGsaaCFt/68Qh1h1vgXS+YN2/eAAAWLVoEJSUlxMTEYPjw4Vi8eDHjdPV
D1Zm9HMfh1atXUFFRkeqPxQdDhw6VbLhRU880vt3IV/Dw8EBycjL69OmDRYsWwdnZGRs3bkRpaSlvWxZUzIz18/O
TOcfH33Pz5s2Rl5cHfX196OvrIy4uDp06dcK9e/d41w8ZAM6fP49z587JDCS0bdsW9+/fZ5Tq/YRQqK8oYIlEIuz
YsQONGzeWnCsrK8OFCxfQvn17VvFq9Pfff2PHjh0IDQ2FtbU1VFVVpc7z8XoxZ84cSauYpUuXYsCAAdi/f79kKSG
f+Pv71+p9IpGId8XNqrtOcxyH3NxcPH36FFu2bGGYrHrz58/H8+fP0bdvX7Rp0wbR0dH47LPPWMeSa8CAAZKfjYy
MkJaWhvz8/A9aBcXCtGnTYG1tjdOnT0NPT4/XWc+ePYuwsDCpAT0zMzNs3ryZN7Pb6pu5c+ciLy8P1tbWKCsrw/n
z59GtWzfWsaSEhIRg586dGD16NADAzc0Ntra2KCsrg4KCAuN01XN1dYWnpyd++eUXdO3aFSKRCPHx8Zg7d67kWSU
+Pp43bWaENvu4NnUM2in9v4l6bn5CL168wP79+7Fjxw4kJyfz6oFfqMvpS0tLsX//fgwYMIB3y5ZqcuXKFRw+fFj
uZiYhISGMUlVvz549Uq/FYjF0dHTQrVs3aGlpMUr135CdnY2EhAQYGRmhU6dOrOPUC5MmTUKrVq2wdOlSbN26Fd9
88w1sbW2RkJCA4cOHY+fOnawjSlFTU8PVq1fRtm1bqKmpITk5GYaGhrhy5QqcnJzkbtbDB/7+/lBQUMDs2bMRGRk
JZ2dnlJWVSQr1fGhKXzGz9P79+2jZsqXUw1GDBg3Qpk0b+Pn58e4BDwDs7e2rPScSiRAREVGHaf6ZoqIiXi8lFKq
qG/1VfGfb2dnxrlg/fPhwqddnzpxBp06dZAqbfLg3qppVHkVFRTRv3hz9+/fHkCFD6iBV7QmpP7aamhouXryIzp0
7Sx1PSkpCnz59eNcDsrLHjx9j7ty5CA8Px5MnT2QGGTfny/Ffd7PQ1a9agd+/esLGxkRzjywBOgwYNcO/ePanrg7K
yMm7duoVWrVoxTFaz169fw9vbG4GBgZLl0YqKipgwYQL8/f2hqqqKa9euAYDMZ54Q8s9RcfMTiIiIQEBAAEJCQqC
vr48RI0ZgxIgRxgRvGqKXbnfmFgsFsxyegBQUVFBeno69PX1WUeplaCgIIwfPx6Ojo4IDQ2Fo6Mjbbt++jdzcXAwbNoy
XI2Kkbgh1kxChKS8vR3l5uaQ/T3BwMGJiYmBsbIxp06bxbhmys7MzrKyssHz5cqipqSElJQX6+voYPXo0ylsvLceT
IEdYRa4XPhXp7e3uEhITQYM0ndvv2bbRt25Z1jH+l4jaZz7PecnJyqn3Qj4uLQ/fu3es4UfU8PDxq9T4+3BvVJmt
5eTmePHmC6OhozJ07V+6KBlYcHBwwf/58wwgwclrQn++usvuLm5QUtLC8eOHWOcsHoDBw5
EdnY2Zs6cKXeGrKurK6Nk0ioG995HJBIhMzPzE6epHQUFBeTm5kr1XK24L6rtfw9Lr1+/RmZmJjiOg5GRkdRqEUL
Ix0fFzY/kwYMH2L17NwICAlBYWIiRI0di5atSE5Olum3wQfR0dGwtbWFoqIioqOja3xvnz596ihV7djb28PLy6v
GJch8YmFhgalTp2LGjBmGaYOnUq9PT0ZGZc8MXz58+xc+dOya6Vpqam8PDw4F3/LqH3SdPR0cGlS5cEVQB
438PbkiVL6ihJ/ZSWlgY7Ozt06dIFERERcHFxwY0bN5CfN4/Y2FgYGRmxjkhIrYjFYujp6aFPnz7o6cP7OzsYGJ
iwjpWrQQGBuLnn3/G7du3AQDt2rXDvHnzMG7cLZLVv3x6xsbHQ1taWOh4bGwtnZ2cUFBSwCfYfcvr0aUyfPp3
55hspKSmSn+/evYvv/8e8+bNk9sf28LCoq7jVSsnJweurq64P48eO86/deWXWzTsm
/JxaLMXDgQKneiidPnoSDg4NuaxY+zPQWmqrtTGpCGwASoaDi5kcwaNAgxMTEYPDgwXBzc4OTkxMUFBSgpKTE2+K
mkB0+fBgLFy6Et7c3unT7c3unTpItN3jE83a8C7ZUE3btxAmzZt0LRpU0RGRsLc3Bzp6elwcHQ9CPjk+joaLi4uEBDQ0P
SKD0xMREFBQU4ceIErwreVUdunz59iqKiIkmPv4KCAqioqEBXV5c3I9GV+fj4QElJSVCbhFSdhV5SUoJ79+5BUVE
RRkZGvLwJEkqxvkJubi5+/fVXJCYmory8HFZWVpgxYywBBpFIhEaNGsHY2Bi9e/fmTZ+sBw8e4MSJE3L
bhfCxf+WbN2+wceNGERZGyt35mI9/d48fP0ZERASiSio6MRFRWFW7duoVmzZpJCZ0X/Xr5Zu3YtFi9ejJkzZ8LW1hY
cxyE2NhabN2/GDz/8AG9vb9NHxYsXrCPK6NixI3r06IFIFff/1VUogoKyvD119/jdjYWFy
/fp1xQvkOHDiALZLVQTdv3sTkyZMxdepuUuLm5MU4oa9asWQgMDISxsbFgNgmR5+XLl3B3d8ewYcN7M+M
pOjoarq6uUFdX532xXsgMDAwkgwtaWlrgOE4yuNC4cWM8efIEhoaGiIyMZN4rKzw8HC4uLjAwMMDNmzfRsWNVgeM4WFlZ8eLBo6qxY8ciNDQUX3zxBBo1aiQNDQX3zxBZo1aybz4LR06VJGyWrvzp07+OGHH7B//36Ul5fz7vu6goGBAXx9fWXf1VUogoKyvD119/jdjYWFy
/s2YNly5bh3r17zzJLJx3EcvvzySzx58gTx58gTnz5/H5cuX4eLigh3rvzBZo1aybz4LR06VJGyWrvzp07+OGHH7B//36Ul5fz7vu6goGBAXx9fWXf1VUogoKyvD119/jdjYWFy
2bVjHqZUvvvgC1tbWLhwodTxn3/+GfHx8Th5uX4eLigh9++IEXCROwfv16Rsnkmz
8+XKp40uXLkVOTg4CAgIVZWzEf55VFWVsa1a9d
klg/evHkTnTt3RnFxMaNkNTMyMsKRI0dkRs8TExPxxRdf8O6BFKgfm4RUuH79OgYPHoysrSzCzWUaQIsVgvpAGnCgc
PHsT27duxY8cOyl5O3fuYOrUqZgyZgyvRoNG/enHnfUBsbGzg5OcHPz0ExEoRxxRdf8O6BFKgfm4RUuH79OgYPHoysrSzCzWUaQIsVgvpAGnCgc
BM2fOwNbWlnWUWnv9+jViYmIQFRWF6OhoXLt2/4ulqkaNGuH69esyG7Hcvn0b5vb5ubGmePmm1404uuqkaNGuH69esyG7Hcvn0b5ubmePPmDaN
k1SspKYGzszMKCwuRkpnaVWQk5OD3377DcXFxXBcUGvXr0Yp5MVHh4Of39/yQqL9u3
bY86cObyfvamlpYWioiKUlpZCRUVFZvk/356jgHctkSIiImBubi51PDU1Ff369cPjx48IjIyNDQ1Ff369cPjx48ZJavfli1bhtevX2Pjx48IjIyNDQ1Ff369cPjx48ZJavfli1bhtevX2PNmjW
so0jR0NBAQkKCTIus27dvw9rampcTcQiRh4qbH1FRURGCgoIQEBCA+Ph4lJWVYe3atfD4Kjdu3aYdCgQVi

5ciVUVFRYx6k3KjZtUlRUxJs3b9CiRQuUl5djzZo1ks1MFi9ezMsNLWxtbTFv3jyZvqa///47fvrpJ1y+fJlNsPd
QUVFBVFSU1I6PABAfHw87OzsUFRUxSvbfEBMTgyFDhuD58+eso0gRWrFeaANOFYyMjHD06FG5u92OGDECmZmZuHT
pEkaMGMG8HYeamhquXbsGIyMjaGlpISYmBh06dEBycjJcXV15V6AH3i19DAoK4l3rlZooKSmhSZMmGDduHOzt7dG
zZ09oaGiwjvVeHTt2xNixY/Htt99KHf/hhx8QFBTEiwGRyr0VK7x69QpjxoyBs7OzVIFeSJ8Z8u+lpqZiyJAhyMn
JQdu2bREUFAQnJycUFhZCLBajsLAQR44c4VXv+k2bNsHb2xtffPGF1LLYI0eOYO3atbwu1O/Zs6fG8xMmTMmTKijJLV
X3X1RRkYGLC0teXdfVF/cuXMHNjY2vLuPa968OX788UeZNgC7du3CwoULqdhNBEORdYD6REVFBZ6envFD09MTNmze
xc+dOrFq1CgsXLkT//v1x4sQJ1hHl+uuvvzvBzmBFTbT0tLk9kpzcXFhlEhaxbhB5Z5+YrEY8+fPx/z581nFqpX
Zs2fDy8sLd+7ckeyyyGhcxXh82bN2PVqlVSD1V8emjq27cvJk+ejJ07d6JLly4QiURISEjA1KlTeT/yf+fOHdy9exe
9e/eGsrKypKypM0CH1XtrchxHB49eoS9e/fycldWKysrpKeny9zEp6en48mTJ9SI1vqPwvUVb9++fAu0927d9GhQwcAwLNnz1hGq9Yvv/yCBQsWWYOvWrbxZofA+zs7Oi6mJwd6
9e5GTk4Ps7GzY2dnB1NSUdbQa+fr6YtSoUbhw4QJsbW0hEokQLAw3izX7Ny5s0xvxYrX27Ztw/bt23nbrod
8WvPnz4e5uTn27duHffv2YfDgwRg0aBB27NgB4F07nFWrVvGqnPjzPPI2HJWrVvGquPnjjz/C399pFwbXWvMCF5y7Aqk
+HTt2xKFDh2Q2gAwKCuL9ikMhu3z5sp8wcBCdFiMvaAtmhlBgrUpTUZXaS5KLbWB8XBbqLaUtkKvDXxn2tbVo4q6sVMRuy4dmpn
vi5vDhw/H6NGjMXLkSNZRaiUzMxPhFDg1Damqq1I19RSGILzfxYrEYjx8/ho6ODusoH0xeX9PK+Nrj9OnTp5gwYQL
Onj0rWRZWUWlqKAQMGYPfu3dDV1WUcUFZeXh9Qq+QSI4xtNnXXGC4sHuacLAMuWrGSQ+JVh0ncXfc7ZSTS/CDo
6OnBwcMCiRYt4MVO9cgE+PT0d8+fPx6+fWNkw2VA8+QW60ENGsUqplyqqqpITU2FoaEh6ygfxNnZGbm5udixY4ekNUR
mT56M5s2b49SpUzh2b49SpUzh58iS+/fZbpKamIjQ2fbT06ExxXBNFyt9nR8fwwfFKBhN5Gbnm5udixY4ekNURSUhI
SFy5cEMzSxwopKSmIjo5GdHQ0LlXNI6FQBBDs7OwQFBbGOJmXNmjWYO3cugHftCxXFsaqnBRyJlyhTMnz8fcXFxjN
Sb0VSvaZNmyIiIgWFhZ4/fo11/NXVER8fL+k5nZGRge7duDeoOgoIBt0ErU1NSQlJQktxWEpaUlXr9+vfoS
6urrk55qoqKhAUZFf84lOnDiBESNGQEkcCRHRw6CJJmsm+TNDNd4eeTwnjd6kN8fZvbTvbTbdPsjHpDTXh7fcSBdAnNrmCb9VCiFU
6OnBwcMCiRYt4MVO9cgE+PT0d8+fPx6+fWNkw2VA8+QW60ENGsUqplyqqqpITU2FoaEh6ygfxNnZGbm5udixY4ekNUR
cjPXr1yM9PR0AYGpqCi8vL8HUBwgBqLj5n1W50Pr06VP4+fnBw7OfnBw8MD5ubmMg9MfJkWHIkCFDAb7/9BkNDQ8T
HxyMvLw8+Pj5Ys2NYb/oIicViaGhovHf2HR8fSIX+0HTr1i3JbpumpqZo164d60jvVnQ2A1NQUycn
JMDQ0xPnz5+Ht7Y0bN26wjxVmgDThVyc3MxbtwhIeHSw0u9O3bF3v37kWpuR80WRmJmJKpISODo6Ms2amZm
J169fw8LCAkVFRSP8TRZg7d66kXYiPHkhIjIPXp5ItBFjGOmZJT1WJcY1SzJHRZNidyGTvyFaEKmcEwP0mn8J8M
Lli1yd+h99eoVBgwYgIKCAqSlpTFIR0jtiMVi5OfnQ0dHh3UUQghpEHz8/Mq3yE7XMrpdLv07ZfhZtjvXWWEHQBjbrh27ZnYNShrQKAF1vjjfX+qDrNC0q8yKo9RT29kWnaJqLwUFFzyK7L7b5MNq8k4kWlgXFatK6Q2WZ6ZcNE10oIMfDB+/QGpCXQlZTMUm8AVyqN0Sqiwk7RJ7Vbtc9XFywJDzTE3aZ8qYXDgO2bG+nAjqEXq+FwLg4XEyO7hYk06dXqE+RvtEWTnw
Lli1yd+h99eoVBgwYgIKCAqSlpTFIR0jtiMVi5OfnQ0dHh3UUQghpEHz8/Mq3yE7XMrpdLv07ZfhZtjvXWWEHQBjbrh27ZnYNShrQKAF1vjjfX+qDrNC0q8yKo9RT29kWnaJqLwUFFzyK7L7b5MNq8k4kWlgXFatK6Q2WZ6ZcNE10oIMfDB+/QGpCXQlZTMUm8AVyqN0Sqiwk7RJ7Vbtc9XFywJDzTE3aZ8qYXDgO2bG+nAjqEXq+FwLg4XEyO7hYk06dXqE+RvtEWTnw

jI6+XmDZo0ADGxsasY9RK69atZZYOAu92d+dr4/yqO9i+ePEC+/fvx44dO5CcnMzL4ubz58+lXlfMDlq8eDFWrFj
BKFX1qm4AOGbMGEZJPoxQ2vUQUp3w8HC4uLjAwMAAN2/eRMeOHZGVlQWO42BlZcU6Xo1OnjyJwMBA2NnZwdPTE71
69YKxsTH09fWxf/9+QRU3u3XrhsmTJ7OOIXiRkZGsI3wQV1dX+Pn5ITg4GMC7+/vs7GwsXLgQI0aMYJyOkNqj4uZ
/WLNmzXDv3j20atUKf//9N65evSpVYHn16pVMTy8+ePPmDTZu3IjIyEi5fd2uXr3KKFn9IdQHpZiYGERGRuKPP/5
Ahw4dZD6/fJltk5KSUuv3WlhYfMIkH67ysvSYmBgMHjwYmzZtgpOTE2+LhxMnTsThw4excOFC1lFq9Pjx4xqvuYq
Kinj69GkdJvowQhpcWL16NWbNmoXNmzejS5cNWbNmoXNmzejS5cuEIlESEhIgJeXF68LyMC79gUBAQEICQmBvr4+RowYgZ07d7KOJVf
VYiEA9O/fHw0bNoS3tzcSExMZpKrerl27WEf4R77//nsUFhYCeLfr9ODBg9GrVy9Jux5C+G7RokXw8fGBn58f1NT
UcPToUejq6sLNzU1q4gUf5efnS3orqqurIz8/HwDQs2dPTJ8+nWW0WBPKoORvLdmjVrMGjQIOj
q6qK4uBh9+vRBbm4uPv/8c14ORBJSHSpu/oc5OTlh4cKF+Omnn/D7779DRUFqpdbSkoK7/r9AYCnpydCQ0PxxRd
fwMbGhvcP0UIk1AclTU1NDBs2jHWM9+rcubub1NkxmPV5SoAf5dtfv311wgKCkLr1q3h4eGBoKAgaGtrs471Xj/++CM
GDx6Ms2fPyt2IhS/L3T777DOkpqZWO/+sxJSWF160KhDK4AADu7u4oKipCt27doKj47laotLQUiocK8PT0hKenp+S
9FQ+qLD148AC7d+9GQEAACgsLMXLkSJSUlODo0aMwMzNjHe+D6ejo8HoVgNAMGDBA8rOhoSHS0tKoXQ8RlPT0dBw
8eBDAu4G84uJiNG7cGH5+fnB1deWA0BBZWVnQ19eHmZZkZgoODYWNjg5MnT0o2rOObqtcGjuPw6tUrqKioYN+
+fQyT1Q9C6zetrq6OmJgYRERE4OrVq5XN/7WMdjZAPQsXN/7AffvgBw4cPR58+fdC4cPR58+fdC4cWPs2bMHDRo0kJwPCAi
Ao6Mjw4TynT59mfOnIGtrS3KbKPWWUB+UhDLgJy4VJSyfuJWYKBBaGeZdWWG8ze6uq+h0oFouho6OYN06hO
VrVu3hoGBAaKjo6vtmCYlrFy5K5Qc/Zoy/ReNYmD/fcs0ysuxx/1rzJBBfdp+DoD+T+cT4ntuHEcELP0eEK8cF
tUCOnqUOuHQPcK3j2pQpnzGuSNUf1Cd7kUbd6KYQWJGVOZmz6xNgBvHCDbbd/BvJ5xXjkyqQaTgHkYOYTvXSYSKp+
AzPTXLOvy5Ik3ut8Obi8M+ZtSNDVz6gcmdV9wVbGiFjKRJF1dNzpFmq+p1IY2QdGNX0xYwvMJgEKzjj9K3JvKkrGWd
fnUSGeN3gtRUnwSugzDRaLpLd2hK0vv71Bp7BhlsQF1HkQNIiXSBzTUdNXpBrJGmwjcLlkyjDd+oLDsfgfzoLy2DhLn
vUEqjVkTb1+XiatIcr/0zaoCtJIsj9WCFFyo8kLLsC60iiz8ZArYgw9+RLk2ko9Z0nZWyjaKhpuK6XEKBV+9jj0pl
U1Yjlh8KDoOb0ix9y/xN78EF+YrHHJNjpb6skfj6G4AY5GPr8X3R9H+nmtaJI1ppgd2TZm/fgXfgPrXsYh3Nx
WrWnY6FJxLHwwfbXBijTUTwZp+UtGxmz1P8iyKdNTXUH08o/RNsJDNJ+RgLpZTiItcBdcpHuMq5DnZxlXtBiUvwTNhO
dwqsVb+YNaSHBFxTlG7Q2XbaZevQ1y2mQ6H3EQjE6Hi+ZAaPB4jYRWegIg9D0Jx0ZAfqCPiDVWVAFyxlo/WHCEgIMB
5P8cG4CtW3lLyVTH69w6CVlEeChGKHt/pKOQ7VXgK8b6dNbvrexWcqQ+BANFz5nW3NyL3DyEHZeSHtDxy3NLhq7LfH
nwhsW1gCcBY5p39OiUqL4eXGNHG4b/Nc4/fbJzYC0f3H8/KWdqGlpI36GcdhGaPjVzpV4Ni9BaMHejBsOwTv4d0Kz8
hIxG/AI/ojLzqNOVaFbXOAO1CxdyqfaGMOLpF9zW8Xj9r/4ImwZkk2Zv6tUZZLjMVwvlbtMoV5ieG8ZUZ/8L3hMSvM
kqVwnHCLBq7pUOMFHROuFdFtjCjd9ywk4HCDYIGuXaUkS9T3MmKzjeAnYe1Ea9H5c0uWdNlYOyxaIDvpHwcH3TE3D
p0pNb4Hi2/gP/rhi2YGz1SFIr4Ls/z4I6nvb6b9IkWSnYGXYdhbPr04QwR2/yxFOMZDOp4UJSV/WL6gZVrfcKeBLXu8
C2ww7Zi3ceSeTNoCDtOMfedUzXQJsCBXD8tkbp6w7oj/6/r+kZLyyWw+KzxmBCN2qPdmt9Nu7rIqSA8qw0dmudfyl
mk5vWJ/vYd3AenknKeDcisFhHZ/+LKNDQjmvIOjDBDVJhQsOxxVQeffoFu+ngvOO6ZXnKRaFf0Y/k7yNk8XZFy1wr
Fs8hN1OWaGwrEJTwmOEaLBGLVGmQL7xnq1P6hnvJTv1oJ/z2sjjLwpTeGIBDBzMfJgvt/8hP2pPE1SUA9U4sBFJ8I2Z
kYjVSJ+QWuyBz9BHlqJ1OJSZD9JmeoYiyk2/lfr4IVPjkzpm3bExUaUftmyHtC5kWSoVhqeI35R8bx7ZbN/7+o8L/
t2nGx/UsavUk97+m61mLYLeL1i2mXxYoWRJ4FqSOSg4TUQvvrZgB8WSzXMPTwubW2u7hyd4ZNRS+M6k/WMxWu8v6WE
uMvQKZF83knrd5OyRkE8yq+ruPzdavvf5+rgP+XVGWAXnoiWrYsbTnQ+pSfMgNpbrZgwqS8tWKIFaw/TjF4YmXPC5m
jJ3Cj3GoLBNwtIK/HnOVs5XOCWUTwAQV7ELg6NbqGhxz1k1h0MxYBL9f5TTSMrMaRHaHvoGHvhfqqwsRNAo2EFOhBa7A

48OHYvXs31NXV37syqHHjxujQoQOmTZv23mXhRNiE2m9aX18f586d410vUEI+FBU3ieC9ePEC+/fvx44dO5CcnEw
zNz8CQ0NDXLlyBdra2lLHCwoKYGVlhczMTEbJ3q+goADx8fF48uSJZBZnBaH0Edq4cSPmzZuHv//+G02bNsW0adO
wcOFCQfU/Jf89eXl5kmtGdnY2duzYgeLiYri4uKBXr16M05G69PjxY3Tu3BkNGjTAzJkz0a5dO4hEImRkZGDTpk0
oLS1FUlISmjVrxjpqvVDRd9Xd3Z11FEL+sYKCAuzcuRPp6ekQiUQwMzODp6cnLwuCHh4e2LBhA9TU1N67aeht29
x+fJlmJub82rjN/LxNW/eHOHh4ejQoQMAQEdHB1euXJEMPN26dQtdu3bFixcvGKaUtWvXLpw9exa7du2iZw0iaFT
cJIIVERGBgIAAhISEQF9fHyNGjMCIESNgaWnJOprgicVi5ObNbmQldXV+r448eP0apVK/z999+MktXs5MmTcHNzQ2F
hIdTU1KSWIotEIsmmIXyUm5uLwMBA7Nq1C9nZ2Rg2bBgmTpyIhw8fYtDDT08P58+dZxxS0xMREqYcmulZ8HKm
pqRgyZAhycnLQtm1bBAUFwcnJCYWFhRCLxSgsLMSRI0d41/oiPDwc4eHhcgdCAgICGKWqWqP+7du4fp06cjNDRUsqG
bSCRC//79sWXLFslMFvLvbdu2DcuWLYObm5Obm5vczVhcXFwYJSOkdhISEEjBgwAAoKyvDxsYGHMchISEBxcXFOH/+vGT
DHqFKS0tD165dD165dUVhYyDoK+YSUlZ7Vx7dq1aluyZGRkoHPnznjz5k0dJ6uZpaUl7t69C47jYPHjz
A7t27ERAQgMLCQowcORK8MFfawQozMzMWMcTvMojyufOnZMaLS8rK0N0N4evQozMzMWMcTvMojyufOnZMaLS8rK0N4eLhkh2E+8vHxgaenJ1auXCmYkceQkBD
s2rUL586dg5mZmZGWbMmIGvvvpKalOTzp07UyHuyHjx5AlGJx6NqKgoaGpqguM4vHjxAvb29jjxAvb29ggKCoKOjg7riII2f/5
8mJubY9++fdi3bx8GDx6MQYMG9gwiy+VDa2J8fK+d4yqctqyKa7sJUmL+2XwBgR1FB1XpjSqgsXwDcw2OudPdmDc2/Hz16+zH1QlbIm7dCU2fqWy1ZRsK3u1+vYCdirp89
Qk3RCijmU3SmJ9MF5r0eFYRsVtPyXTMK6cyhgdxQyfsC0yV1ZBZmN0lXwiegKTSWWL6nP+QYyhtOK4LcWYiZJVm0TAEK6E+89vdKV93Ui3+gyPbD2+xm8u3H6Qn6E9b+DzEHb8LzwHTEAh+6aSBw4SUm9XmABy8d+Sbg
2T+TL6
AANv1ZDkDJfAGyK6LIEu48abU3lJu5R+nVr6j/cxwWvgWq58+KUEk0w+YKi3FwPTE/lNVDK8/x6Kb8mIuEgsPh/3PbSWm6W4yh0/nlnXV3bK/1Pb20JjYwb3Jj16cdnJ94kkV/MdE3/JjJkXz+8EmhG+kDKbRiWYUXvv6+Wy3x+Ix7Iwpvp9zaQKnnpfP8+KzlU2Hwb/qOGYK8X94ZbD3Uy4+GYNW7ABkz6tBLpjQSw8VOXgzNLB0J0Dz50uUXrBk3rx+VhRXe5zUVt3z/hkEU0BRtvpdXYl2c5lAGZSCLzYzqA7hW6HaZDkuVgh3MDHnuWvv6uGHe68J+LSK+0v2H5Q3BHPZsLMaPaSbr4DCWxB7bmRFg8Sb6KX5VoAE1m2y91sIfA8YpkDkUEO2Ldc6iCDMD2uu0NeuUBpd7mwZk6eQ7Z48C5mlzMzL9GV6NRJMAydW7NhLT1v5ZsThRpUjHNc7BSgEiC9/zmTfQlrPRcS1COuSgbL4OlsGVb3VqrZ0nx+LFcsRmGqZ9IAsZfkbYHTzRGDrR3EkLi7Ci2mLZYhXMCzH3J5MV0xqMN9V4Z2hpMDuxcl9hsNvXN9CQmI4HALKJkSc5tvMIe4sdPTvebcXStAuIcMF0aDAMB4f0XYzKCFiXrG9AHbzz+uR0cBe/mbyRV5GBi9ydHI+KHM05wJjGdEFYbuWkxdzv2c2Za2tmaOVRwBdtbgwKZeW9WSWj6Mq1Q5KJGaVEVuNgYVK7e4FzYVKQoW67UPxF/x8oaQS+sDEdpZELkH0WDYEVOqnqzfsjZyqgkU3XWqu4b3cEIxwedPqjdR2h64RqUEMlytjMnx4SiyvRwXOLxSHX1UcDJ8qn3KdHvWyKRjwWSMo2EQNPMK6BH0WpLu0cQkLTfBNEFQ3tYNzt8jNk7+nsP9qvkxTEf7RFgUKcc8dVLPyF6hgtcjTDycCUAR7KLYeNFVd9VJBPIgqrKGY8ahmZiT9aH5Rl2jaJ0i3+Z6NWGkjL08kwkmvM2cjG9KxdfVyWnfqoYwdrVtsMlnZ6u7q2lD5hdoYm1Cu7/UqZ9/5WxGQp9aw3bkahwlxfVNLT8eKX3NmYqJZMbMgbsD03xXfBvn2b6b0fiRk6O+LZOsZpkqXqUYEdwW4OxaaCyDwyVt9lu7DdtfAdmLQ87iL1Zlb2zrT8o9v/vP2k5H6pDn6Zv2CoWaGvY0bWkYJvPTFRnYlOWTUHnYEXGdoK8sGk0kXwLemP8f6D+3j5y9VEY8+fgMPpx3dT3ZMVHHdRYw94+E2nn4EAUKZ4P0RfZGmgBeWkfhXVpafZx6oVp/5Bn8zeyTmP6AbZG3APG0qYwuUrw0Ap65+ldfrI9bBDGdKCE8QPOG9ZsmiJbFM/GRDKv+w4rp6SF4KKOMMPnGBSdCGmUhpeQQpYfHbCw9FHPr/uxAVAN+KKhPcsjJnCvpB1xAPIYUl9qAbA1MddoPCg9dHhPejc8

48OHYvXs31NXV37syqHHjxujQoQOmTZv23mXhRNiE2m9aX18f586d410vUEI+FBU3ieC9ePEC+/fvx44dO5CcnEw
zNz8CQ0NDXLlyBdra2lLHCwoKYGVlhczMTEbJ3q+goADx8fF48uSJZBZnBaH0Edq4cSPmzZuHv//+G02bNsW0adO
wcOFCQfU/Jf89eXl5kmtGdnY2duzYgeLiYri4uKBXr16M05G69PjxY3Tu3BkNGjTAzJkz0a5dO4hEImRkZGDTpk0
oLS1FUlISmjVrxjpqvVDRd9Xd3Z11FEL+sYKCAuzcuRPp6ekQiUQwMzODp6cnLwuCHh4e2LBhA9TU1N67aeht29
x+fJlmJub82rjN/LxNW/eHOHh4ejQoQMAQEdHB1euXJEMPN26dQtdu3bFixcvGKaUtWvXLpw9exa7du2iZw0iaFT
cJIIVERGBgIAAhISEQF9fHyNGjMCIESNgaWnJOprgicVi5ObmQldXV+r448eP0apVK/z999+MktXs5MmTcHNzQ2F
hIdTU1KSWIotEIsmmIXyUm5uLwMBA7Nq1C9nZ2Rg2bBgmTpyIhw8fYtDDT08P58+dZxxS0xMREqYcmulZ8HKm
pqRgyZAhycnLQtm1bBAUFwcnJCYWFhRCLxSgsLMSRI0d41/oiPDwc4eHhcgdCAgICGKWqWqP+7du4fp06cjNDRUsqG
bSCRC//79sWXLFslMFvLvbdu2DcuWLYObm5vczVhcXFwYJSOkdhISEEjBgwAAoKyvDxsYGHMchISEBxcXFOH/+vGT
DHqFKS0tD165dUVhYyDoK+YSUlZ7Vx7dq1aluyZGRkoHPnznjz5k0dJ6uZpaUl7t69C47jBLXXAiFV0VpHIigPHjz
A7t27ERAQgMLCQowcORIlJSU4evQozMzMWMcTvMojyufOnZMaLS8rK0N0N4eLhkh2E+8vHxgaenJ1auXCmYkceQkBD
s2rUL586dg5mZmZGWbMmIGvvvpKalOTzp07UyHuyX3jy5AlGjx6NqKgoaGpqguM4vHjxAvb29ggKCoKOjg7riII2f/5
8mJubY9++fdi3bx8GDx6MQYMGyceOHQDeNapftWoVr4qbvr6+8PPPzg7W1NfT09Kh5/idgYGCAs2fPIj8/H3fu3AE
AGBsbo0mTJoyT1T/Tp08HALn9qfmavp4oOjt+Pj8EviikmlgNGVpd4LSFSAwuI/xcPi1LAjbDLP6IXi6hfGZK6t0FX3AeBp
8YkLtN82n+zNC/g2auUkEY9CgQYiJicHgwYPh5uYGJycnCgoQElJCcnJ3qGPu1rnjVyV/TRymuazVBW37ODjCLjBpaTRiCumrQl
55RdJo2y+UVVVRWpqqqBmBGloaGD06NkSunbtKvc9xXFWL16NfWG/IdGjRFu3fvYu/evYRA1NQXwbhbFhAk
TYGxszLvNboSmadOmWGp+KcyVtr7VX36BCLIGmxsmfPk+QvGvIQrnCbkVnD+bVV+I6uNKi5RgsAx06W7OFI7vmlNVP21MXxHmO7jd4EjSOKGCNYWCsDOBVvxNDGZsOVpQHBBL9E3vwS2DEzK8fhuwNHPXR8cEw2qxMhLHE8E4tAsKs+I+M6RKYHZB0u+P6XPmsGQxW5dv0/cSUQ0ZRrZfiwTJ8XSqqXYyX2XOhtgBxXN+TJmgY3b/XLSfjHI0UEFBTsEpEPNSOIeWVUPtgdh87JYOtjrAeRuyZ4IqLhZcNz8YvY7BbC0Ge/QuLwNJgnH5U/E/WGJ+4xXtAGMuR5X7mPdAH5lARHbdPSMixuQ2Dy5e0ssvDh/JNlsOwuGWJrBhfLtEqN0KcD8jFcnYNNHdR0BJJyh4Dp62R0JRsAZ+/vD4W9uQJ5+4k/fQaAzHDNpEhtBLZ0MngZrVHbWC8Y/rEmzTkgq3U0JmhZAmTIYqV/2o2/sTbQR6sb

UkZGRMDc3R3p6OhwcHPDo0SPWEQkh5F958OABTpw4Ibf1xtq1axmlIoQQIgQDBw6EoqIiFixYgH379uHUqVNwdHT
Ejh07ALzr+52YmIi4uDjGSQmpWVFREYKCghAQEID4+HiUlZVh7dq18PT0hJqaGut4hNQaFTcJITJatWqFM2fOwNz
cHJ06dcLChQsxZswYXL58GU5OTnjx4gXriIQQ8o+Fh4fDxcUFBgYGuHnzJjp27IisrCxwHAcrKytEREScwjkgIIYT
HmjZtioiICFhYWOD169dQV1dHfHw8rK2tAQAZGRno3r07CgoK2AYl5APcvHkTO3fuxN69e1FQUID+/fvjxIkTrGM
RUivVb+9GCPnP6tWrF0JDQwEAI0eOhJeXFyZPnowxY8agb9++jNMRQsi/s2jRIvj4+OD69eto1KgRjh49ipycHPT
p0wdffvkl63iEEEJ4Lj8/H82bNwcANG7cGKqqqmjSpInkvJaWFl69esUqHiH/iImJCVavXo0HDx7g4MGDrOMQ8kF
o5iYhREZ+fj7evHmDFi1aoLy8HGvWrEFMTAyMjY2xePFi6ldICBE0NTU1XLt2DUZGRtDS0kJMTAw6dOiA5ORkuLq
6Iisri3VEQgghPCYWi/H48WPo6OgAePe9kpKSAgMDAwDA48eP0aJFC5SVlbGMSQgh/xmKrAMQQvin8sizWCzG/Pn
zMX/+fIaJCCHk41FVVcXbt28BAC1atMDdu3fRoUMHAMCzZ89YRiOEECIQ7u7uaNiwIQDgzZs3mDZtGlRVVQFA8h1
DCCGkblBxkxAiV3l5Oe7cuYMnT56gvLxc6lzv3r0ZpSKEkH+ve/fuiI2NhZmZGZydneHj44PU1FSEhISge/furOM
RQgjhuQkTJki9/uqrr2TeM378+LqKQwgh/3m0LJ0QIiMuLg5jx47F/fv3UfUSIRKJaIkNIUTQMjMz8fr1a1hYWKC
oqAhz586VtN7w9/eHvr4+64iEEEIIYSQWqLiJiFERufOndGuXTv4+vpCT08PIpFI6ryGhgajZIQQQgghhBBCCCH
/Q8VNQogMVVVVJCcnw9jYmHUUQgj56HJyciASidCyZUsAQHx8PA4cOAAzMzNMmTKFcTpCCCGEEELIhxCzDkAI4Z9
u3brhzp07rGMQQsgnMXbsWERGRgIAcnNz0a9fP8THx+Pbb7+Fn58f43SEEEIIIYSQD0EbChFCZMyaNQs+Pj7Izc2
Fubk5lJSUpM5bWFgwSkYIIf/e9evXYWNjAwAIDg6Gubk5YmNjcf78eUybNg1LlixhnJAQQgghhBBSW1TcJITIGDF
iBADA09NTckwkEoHjONpQiBAieCUlJWjYsCEAICwsDC4uLgCA9u3b49GjRyyjEUIIIYQQQj4QFTcJITLu3bvHOgI
hhHwyHTp0wNatW+Hs7IzQ0FAsX74cAPDw4UNoa2szTkcIIYQQQgj5ELShECGEEEL

CCCGEEEI+ABU3CSGEEELqqYruQ1WXmv8T7u7uuHPnDuLi4gAAAQEBGDlypNwZoYQQQgghhNQVKm4SQgghhNRTbdu
2hUgkQnp6+r/+f0tXVxdDhgzBrl278OTJE5w5cwaenp4fISUhhBBCCCH/HBU3CSGEEELqqSZNmmDAgAHYvHkzCgs
LZc4XFBTIHGvQoAEAoKysTObcpEmTEBQUhG3btsHIyAi2trYfPTMhBBCCCEfgoqbhBBCCCH12JYtW1BWVgYbGxs
cPXoUt2/fRnp6OjZs2IDPP/9c5v26urpQVlbG2bNn8fjxY7x48UJybbsCAAdDQ0MAPP/xAGwkRQgghhBBeoOImIYQ
QQkg9ZmBggKtXr8Le3h4+Pj7o2LEj+vfvj/DwcPz67Dz6668y71dUVMSGDRuwbds2tGjRAq6urpJzYrEY7u7uuKCsrw/j
x4+vyP4MQQgghhBC5RFxFp3lCCCGEEELeY/LkyXj8+DFOnDjBOgohhBBCCCFQZB2AEEIIIYTw34sXL3DlyhXs378
fx48fZx2HEEIIIYQQAFTcJIQQQggghteDq6or4+HhMnToV/fv3Zx2HEEIIIYQQALQsnRBCCCCGEEEIIIYQQIlC0oRA
hhBBCCCGEEEIIIUSQqLhJCCGEEEIIIYQQQggRJCpuEkIIIYQQQggBBCBImKm4QQQQgghhBBCCCGEEEGi4iYhhBB
CCCGEEEIIIUSQqLhJCCGEEEIIIYQQQggRJCpuEkIIIYQQQggBBCBImKm4QQQQgghhBBCCCGEEEGi4iYhhBBCCCG
EEEIIIUSQ/g9Uc2riyD+4fAAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 1500x800 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plt.figure(figsize=(15,8))\n",
"sns.barplot(x='City',   y='AQI',   data=city_median_AQI_per_year,hue='year').set(title
='City vs Median AQI per year')\n",
"plt.xticks(rotation=90)\n",
"plt.legend(loc=(1.01, 1))\n",
"plt.show()"
]
},
{
"cell_type": "code",
"execution_count": 25,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"   .dataframe tbody tr th:only-of-type {\n",
"   vertical-align: middle;\n",
"   }\n",
"\n",
"   .dataframe tbody tr th {\n",
"   vertical-align: top;\n",
"   }\n",
"\n",
"   .dataframe thead th {\n",
"   text-align: right;\n",
"   }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
" <thead>\n",
"   <tr style=\"text-align: right;\">\n",
"   <th></th>\n",
"   <th>City</th>\n",

```
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>...</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>Particulate Matter</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"      <td>0.02</td>\n",
"      <td>224.85</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
```

```
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>...</td>\n",
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"      <td>12.95</td>\n",
"      <td>119.34</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>...</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"      <td>25.45</td>\n",
"      <td>170.21</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>...</td>\n",
```

```
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"      <td>15.57</td>\n",
"      <td>121.16</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>...</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"      <td>28.68</td>\n",
"      <td>208.66</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
```

```
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29526</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>15.02</td>\n",
"      <td>50.94</td>\n",
"      <td>7.68</td>\n",
"      <td>25.06</td>\n",
"      <td>19.54</td>\n",
"      <td>12.47</td>\n",
"      <td>0.47</td>\n",
"      <td>8.55</td>\n",
"      <td>23.30</td>\n",
"      <td>...</td>\n",
"      <td>0.73</td>\n",
"      <td>41.0</td>\n",
"      <td>Good</td>\n",
"      <td>131.18</td>\n",
"      <td>46.89</td>\n",
"      <td>15.04</td>\n",
"      <td>97.07</td>\n",
"      <td>65.96</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29527</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>24.38</td>\n",
"      <td>74.09</td>\n",
"      <td>3.42</td>\n",
"      <td>26.06</td>\n",
"      <td>16.53</td>\n",
"      <td>11.99</td>\n",
"      <td>0.52</td>\n",
"      <td>12.72</td>\n",
"      <td>30.14</td>\n",
"      <td>...</td>\n",
"      <td>0.38</td>\n",
"      <td>70.0</td>\n",
"      <td>Satisfactory</td>\n",
"      <td>156.99</td>\n",
"      <td>46.19</td>\n",
"      <td>3.33</td>\n",
"      <td>101.38</td>\n",
"      <td>98.47</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"      <th>29528</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>22.91</td>\n",
"      <td>65.73</td>\n",
"      <td>3.45</td>\n",
"      <td>29.53</td>\n",
"      <td>18.33</td>\n",
"      <td>10.71</td>\n",
"      <td>0.48</td>\n",
"      <td>8.42</td>\n",
"      <td>30.96</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>68.0</td>\n",
"      <td>Satisfactory</td>\n",
"      <td>151.14</td>\n",
"      <td>39.40</td>\n",
"      <td>0.02</td>\n",
"      <td>101.88</td>\n",
"      <td>88.64</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29529</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>16.64</td>\n",
"      <td>49.97</td>\n",
"      <td>4.05</td>\n",
"      <td>29.26</td>\n",
"      <td>18.80</td>\n",
"      <td>10.03</td>\n",
"      <td>0.52</td>\n",
"      <td>9.84</td>\n",
"      <td>28.30</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>54.0</td>\n",
"      <td>Satisfactory</td>\n",
"      <td>129.27</td>\n",
"      <td>38.14</td>\n",
"      <td>0.00</td>\n",
"      <td>100.80</td>\n",
"      <td>66.61</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29530</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>15.00</td>\n",
"      <td>66.00</td>\n",
"      <td>0.40</td>\n",
"      <td>26.85</td>\n",
```

```
"      <td>14.05</td>\n",
"      <td>5.20</td>\n",
"      <td>0.59</td>\n",
"      <td>2.10</td>\n",
"      <td>17.05</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>50.0</td>\n",
"      <td>Good</td>\n",
"      <td>128.09</td>\n",
"      <td>19.15</td>\n",
"      <td>0.00</td>\n",
"      <td>66.24</td>\n",
"      <td>81.00</td>\n",
"      <td>2020</td>\n",
"      <td>7</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>29531 rows × 22 columns</p>\n",
"</div>"
],
"text/plain": [
"               City  PM2.5    PM10     NO    NO2    NOx    NH3     CO    SO2  \\\n",
"0         Ahmedabad  73.24  141.54   0.92  18.22  17.15  26.64   0.92  27.64   \n",
"1         Ahmedabad  73.24  141.54   0.97  15.69  16.46  26.64   0.97  24.55   \n",
"2         Ahmedabad  73.24  141.54  17.40  19.30  29.70  26.64  17.40  29.07   \n",
"3         Ahmedabad  73.24  141.54   1.70  18.48  17.97  26.64   1.70  18.59   \n",
"4         Ahmedabad  73.24  141.54  22.10  21.42  37.76  26.64  22.10  39.33   \n",
"...             ...    ...     ...    ...    ...    ...    ...    ...    ...   \n",
"29526  Visakhapatnam  15.02   50.94   7.68  25.06  19.54  12.47   0.47   8.55   \n",
"29527  Visakhapatnam  24.38   74.09   3.42  26.06  16.53  11.99   0.52  12.72   \n",
"29528  Visakhapatnam  22.91   65.73   3.45  29.53  18.33  10.71   0.48   8.42   \n",
"29529  Visakhapatnam  16.64   49.97   4.05  29.26  18.80  10.03   0.52   9.84   \n",
"29530  Visakhapatnam  15.00   66.00   0.40  26.85  14.05   5.20   0.59   2.10   \n",
"\n",
"           O3  ...  Xylene    AQI    AQI_Bucket  Vehicular Pollution  \\\n",
"0      133.36  ...    0.00  209.0          Poor               278.63   \n",
"1       34.06  ...    3.77  209.0          Poor               275.51   \n",
"2       30.70  ...    2.25  209.0          Poor               325.22   \n",
"3       36.08  ...    1.00  209.0          Poor               281.27   \n",
"4       39.31  ...    2.78  209.0          Poor               344.80   \n",
"...       ...  ...     ...    ...           ...                  ...   \n",
"29526   23.30  ...    0.73   41.0          Good               131.18   \n",
"29527   30.14  ...    0.38   70.0  Satisfactory               156.99   \n",
"29528   30.96  ...    0.00   68.0  Satisfactory               151.14   \n",
"29529   28.30  ...    0.00   54.0  Satisfactory               129.27   \n",
"29530   17.05  ...    0.00   50.0          Good               128.09   \n",
"\n",
"      Industrial Pollution  Organic Pollutants  Inorganic Pollutants  \\\n",
"0                   161.02                0.02                224.85   \n",
"1                    71.56               12.95                119.34   \n",
"2                    85.22               25.45                170.21   \n",
"3                    70.24               15.57                121.16   \n",
```

```
"4                   107.32               28.68                208.66   \n",
"...                     ...                 ...                   ...   \n",
"29526                46.89               15.04                 97.07   \n",
"29527                46.19                3.33                101.38   \n",
"29528                39.40                0.02                101.88   \n",
"29529                38.14                0.00                100.80   \n",
"29530                19.15                0.00                 66.24   \n",
"\n",
"       Particulate Matter  year  month  \n",
"0                  214.78  2015      1  \n",
"1                  214.78  2015      1  \n",
"2                  214.78  2015      1  \n",
"3                  214.78  2015      1  \n",
"4                  214.78  2015      1  \n",
"...                   ...   ...    ...  \n",
"29526               65.96  2020      6  \n",
"29527               98.47  2020      6  \n",
"29528               88.64  2020      6  \n",
"29529               66.61  2020      6  \n",
"29530               81.00  2020      7  \n",
"\n",
"[29531 rows x 22 columns]"
]
},
"execution_count": 25,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df"
]
},
{
"cell_type": "code",
"execution_count": 26,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"categorical_attributes ['City', 'AQI_Bucket']\n",
"<class 'pandas.core.frame.DataFrame'>\n",
"RangeIndex: 29531 entries, 0 to 29530\n",
"Data columns (total 21 columns):\n",
" #   Column                Non-Null Count  Dtype  \n",
"---  ------                --------------  -----  \n",
" 0   City                  29531 non-null  int32  \n",
" 1   PM2.5                 29531 non-null  float64\n",
" 2   PM10                  29531 non-null  float64\n",
" 3   NO                    29531 non-null  float64\n",
" 4   NO2                   29531 non-null  float64\n",
" 5   NOx                   29531 non-null  float64\n",
```

```
" 6   NH3                  29531 non-null  float64\n",
" 7   CO                   29531 non-null  float64\n",
" 8   SO2                  29531 non-null  float64\n",
" 9   O3                   29531 non-null  float64\n",
" 10  Benzene              29531 non-null  float64\n",
" 11  Toluene              29531 non-null  float64\n",
" 12  Xylene               29531 non-null  float64\n",
" 13  AQI                  29531 non-null  float64\n",
" 14  Vehicular Pollution  29531 non-null  float64\n",
" 15  Industrial Pollution 29531 non-null  float64\n",
" 16  Organic Pollutants   29531 non-null  float64\n",
" 17  Inorganic Pollutants 29531 non-null  float64\n",
" 18  Particulate Matter   29531 non-null  float64\n",
" 19  year                 29531 non-null  int64  \n",
" 20  month                29531 non-null  int64  \n",
"dtypes: float64(18), int32(1), int64(2)\n",
"memory usage: 4.6 MB\n"
]
}
],
"source": [
"from sklearn.preprocessing import LabelEncoder\n",
"categorical_attributes = list(df.select_dtypes(include=['object']).columns)\n",
"print(\"categorical_attributes\",categorical_attributes)\n",
"le=LabelEncoder()\n",
"df['City']=le.fit_transform(df['City'].astype(str))\n",
"final_df = df.drop(['AQI_Bucket'], axis=1)\n",
"final_df.info()"
]
},
{
"cell_type": "code",
"execution_count": 27,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
```

```
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>...</th>\n",
"      <th>Xylene</th>\n",
"      <th>AQI</th>\n",
"      <th>AQI_Bucket</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>Particulate Matter</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>...</td>\n",
"      <td>0.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"      <td>0.02</td>\n",
"      <td>224.85</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
```

```
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>...</td>\n",
"      <td>3.77</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"      <td>12.95</td>\n",
"      <td>119.34</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>...</td>\n",
"      <td>2.25</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"      <td>25.45</td>\n",
"      <td>170.21</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
```

```
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>...</td>\n",
"      <td>1.00</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"      <td>15.57</td>\n",
"      <td>121.16</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>...</td>\n",
"      <td>2.78</td>\n",
"      <td>209.0</td>\n",
"      <td>Poor</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"      <td>28.68</td>\n",
"      <td>208.66</td>\n",
"      <td>214.78</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>5 rows × 22 columns</p>\n",
"</div>"
],
"text/plain": [
"   City  PM2.5    PM10     NO    NO2    NOx    NH3     CO    SO2      O3  ...  \\\n",
"0     0  73.24  141.54   0.92  18.22  17.15  26.64   0.92  27.64  133.36  ...   \n",
"1     0  73.24  141.54   0.97  15.69  16.46  26.64   0.97  24.55   34.06  ...   \n",
"2     0  73.24  141.54  17.40  19.30  29.70  26.64  17.40  29.07   30.70  ...   \n",
"3     0  73.24  141.54   1.70  18.48  17.97  26.64   1.70  18.59   36.08  ...   \n",
"4     0  73.24  141.54  22.10  21.42  37.76  26.64  22.10  39.33   39.31  ...   \n",
"\n",
"   Xylene    AQI  AQI_Bucket  Vehicular Pollution Industrial Pollution  \\\n",
"0    0.00  209.0        Poor               278.63               161.02   \n",
"1    3.77  209.0        Poor               275.51                71.56   \n",
```

```
"2    2.25  209.0        Poor               325.22               85.22   \n",
"3    1.00  209.0        Poor               281.27               70.24   \n",
"4    2.78  209.0        Poor               344.80              107.32   \n",
"\n",
"   Organic Pollutants  Inorganic Pollutants  Particulate Matter  year  month  \n",
"0                0.02                224.85              214.78  2015      1  \n",
"1               12.95                119.34              214.78  2015      1  \n",
"2               25.45                170.21              214.78  2015      1  \n",
"3               15.57                121.16              214.78  2015      1  \n",
"4               28.68                208.66              214.78  2015      1  \n",
"\n",
"[5 rows x 22 columns]"
]
},
"execution_count": 27,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"df.head()"
]
},
{
"cell_type": "code",
"execution_count": 28,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"<class 'pandas.core.frame.DataFrame'>\n",
"RangeIndex: 29531 entries, 0 to 29530\n",
"Data columns (total 21 columns):\n",
" #   Column                Non-Null Count  Dtype  \n",
"---  ------                --------------  -----  \n",
" 0   City                  29531 non-null  int32  \n",
" 1   PM2.5                 29531 non-null  float64\n",
" 2   PM10                  29531 non-null  float64\n",
" 3   NO                    29531 non-null  float64\n",
" 4   NO2                   29531 non-null  float64\n",
" 5   NOx                   29531 non-null  float64\n",
" 6   NH3                   29531 non-null  float64\n",
" 7   CO                    29531 non-null  float64\n",
" 8   SO2                   29531 non-null  float64\n",
" 9   O3                    29531 non-null  float64\n",
" 10  Benzene               29531 non-null  float64\n",
" 11  Toluene               29531 non-null  float64\n",
" 12  Xylene                29531 non-null  float64\n",
" 13  AQI                   29531 non-null  float64\n",
" 14  Vehicular Pollution   29531 non-null  float64\n",
" 15  Industrial Pollution  29531 non-null  float64\n",
" 16  Organic Pollutants    29531 non-null  float64\n",
```

```
" 17  Inorganic Pollutants  29531 non-null  float64\n",
" 18  Particulate Matter    29531 non-null  float64\n",
" 19  year                  29531 non-null  int64  \n",
" 20  month                 29531 non-null  int64  \n",
"dtypes: float64(18), int32(1), int64(2)\n",
"memory usage: 4.6 MB\n"
]
}
],
"source": [
"final_df.info()"
]
},
{
"cell_type": "code",
"execution_count": 29,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
```

```
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"      <td>0.02</td>\n",
"      <td>224.85</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"      <td>12.95</td>\n",
"      <td>119.34</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
```

```
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"      <td>25.45</td>\n",
"      <td>170.21</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"      <td>15.57</td>\n",
"      <td>121.16</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
```

```
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"      <td>28.68</td>\n",
"      <td>208.66</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"   City  PM2.5    PM10     NO    NO2    NOx    NH3     CO    SO2      O3  \\\n",
"0     0  73.24  141.54   0.92  18.22  17.15  26.64   0.92  27.64  133.36   \n",
"1     0  73.24  141.54   0.97  15.69  16.46  26.64   0.97  24.55   34.06   \n",
"2     0  73.24  141.54  17.40  19.30  29.70  26.64  17.40  29.07   30.70   \n",
"3     0  73.24  141.54   1.70  18.48  17.97  26.64   1.70  18.59   36.08   \n",
"4     0  73.24  141.54  22.10  21.42  37.76  26.64  22.10  39.33   39.31   \n",
"\n",
"   Benzene  Toluene  Xylene  Vehicular Pollution  Industrial Pollution  \\\n",
"0     0.00     0.02    0.00               278.63                161.02   \n",
"1     3.68     5.50    3.77               275.51                 71.56   \n",
"2     6.80    16.40    2.25               325.22                 85.22   \n",
"3     4.43    10.14    1.00               281.27                 70.24   \n",
"4     7.01    18.89    2.78               344.80                107.32   \n",
"\n",
"   Organic Pollutants  Inorganic Pollutants  year  month  \n",
"0                0.02                224.85  2015      1  \n",
"1               12.95                119.34  2015      1  \n",
"2               25.45                170.21  2015      1  \n",
"3               15.57                121.16  2015      1  \n",
"4               28.68                208.66  2015      1  "
]
},
"execution_count": 29,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X_train = final_df.loc[final_df['year'] <= 2018][['City', 'PM2.5', 'PM10', 'NO', 'NO2',
'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'Vehicular Pollution',
'Industrial   Pollution',   'Organic   Pollutants',   'Inorganic   Pollutants',   'year',
'month']]\n",
"X_train.head()"
]
},
{
"cell_type": "code",
"execution_count": 30,
"metadata": {},
"outputs": [
```

```
{
"data": {
"text/plain": [
"0    209.0\n",
"1    209.0\n",
"2    209.0\n",
"3    209.0\n",
"4    209.0\n",
"Name: AQI, dtype: float64"
]
},
"execution_count": 30,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"y_train = final_df.loc[final_df['year'] <= 2018].AQI\n",
"y_train.head()"
]
},
{
"cell_type": "code",
"execution_count": 31,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"(17439, 19)\n",
"(17439,)\n"
]
}
],
"source": [
"print(X_train.shape)\n",
"print(y_train.shape)"
]
},
{
"cell_type": "code",
"execution_count": 32,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
```

```
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>110.71</td>\n",
"      <td>141.54</td>\n",
"      <td>63.03</td>\n",
"      <td>111.56</td>\n",
"      <td>100.04</td>\n",
"      <td>26.64</td>\n",
"      <td>63.03</td>\n",
"      <td>80.15</td>\n",
"      <td>57.12</td>\n",
"      <td>4.08</td>\n",
"      <td>32.33</td>\n",
"      <td>6.93</td>\n",
"      <td>616.55</td>\n",
"      <td>180.61</td>\n",
"      <td>43.34</td>\n",
"      <td>501.57</td>\n",
"      <td>2019</td>\n",
```

```
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>147.57</td>\n",
"      <td>141.54</td>\n",
"      <td>59.56</td>\n",
"      <td>107.46</td>\n",
"      <td>129.87</td>\n",
"      <td>26.64</td>\n",
"      <td>59.56</td>\n",
"      <td>47.70</td>\n",
"      <td>48.23</td>\n",
"      <td>4.10</td>\n",
"      <td>32.34</td>\n",
"      <td>6.99</td>\n",
"      <td>672.20</td>\n",
"      <td>139.36</td>\n",
"      <td>43.43</td>\n",
"      <td>479.02</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
"      <td>131.50</td>\n",
"      <td>141.54</td>\n",
"      <td>119.68</td>\n",
"      <td>75.82</td>\n",
"      <td>88.04</td>\n",
"      <td>26.64</td>\n",
"      <td>119.68</td>\n",
"      <td>55.29</td>\n",
"      <td>43.25</td>\n",
"      <td>4.09</td>\n",
"      <td>32.42</td>\n",
"      <td>7.00</td>\n",
"      <td>702.90</td>\n",
"      <td>142.05</td>\n",
"      <td>43.51</td>\n",
"      <td>528.40</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0</td>\n",
"      <td>102.12</td>\n",
"      <td>141.54</td>\n",
"      <td>57.92</td>\n",
"      <td>95.29</td>\n",
"      <td>54.93</td>\n",
```

```
"      <td>26.64</td>\n",
"      <td>57.92</td>\n",
"      <td>69.02</td>\n",
"      <td>51.71</td>\n",
"      <td>4.09</td>\n",
"      <td>32.38</td>\n",
"      <td>6.98</td>\n",
"      <td>536.36</td>\n",
"      <td>164.18</td>\n",
"      <td>43.45</td>\n",
"      <td>413.43</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0</td>\n",
"      <td>115.00</td>\n",
"      <td>141.54</td>\n",
"      <td>63.86</td>\n",
"      <td>111.04</td>\n",
"      <td>61.99</td>\n",
"      <td>26.64</td>\n",
"      <td>63.86</td>\n",
"      <td>86.65</td>\n",
"      <td>59.25</td>\n",
"      <td>4.12</td>\n",
"      <td>32.43</td>\n",
"      <td>6.97</td>\n",
"      <td>583.93</td>\n",
"      <td>189.42</td>\n",
"      <td>43.52</td>\n",
"      <td>473.29</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
```

```
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12087</th>\n",
"      <td>25</td>\n",
"      <td>15.02</td>\n",
"      <td>50.94</td>\n",
"      <td>7.68</td>\n",
"      <td>25.06</td>\n",
"      <td>19.54</td>\n",
"      <td>12.47</td>\n",
"      <td>0.47</td>\n",
"      <td>8.55</td>\n",
"      <td>23.30</td>\n",
"      <td>2.24</td>\n",
"      <td>12.07</td>\n",
"      <td>0.73</td>\n",
"      <td>131.18</td>\n",
"      <td>46.89</td>\n",
"      <td>15.04</td>\n",
"      <td>97.07</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12088</th>\n",
"      <td>25</td>\n",
"      <td>24.38</td>\n",
"      <td>74.09</td>\n",
"      <td>3.42</td>\n",
"      <td>26.06</td>\n",
"      <td>16.53</td>\n",
"      <td>11.99</td>\n",
"      <td>0.52</td>\n",
"      <td>12.72</td>\n",
"      <td>30.14</td>\n",
"      <td>0.74</td>\n",
"      <td>2.21</td>\n",
"      <td>0.38</td>\n",
"      <td>156.99</td>\n",
"      <td>46.19</td>\n",
"      <td>3.33</td>\n",
"      <td>101.38</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12089</th>\n",
"      <td>25</td>\n",
"      <td>22.91</td>\n",
"      <td>65.73</td>\n",
"      <td>3.45</td>\n",
```

```
"      <td>29.53</td>\n",
"      <td>18.33</td>\n",
"      <td>10.71</td>\n",
"      <td>0.48</td>\n",
"      <td>8.42</td>\n",
"      <td>30.96</td>\n",
"      <td>0.01</td>\n",
"      <td>0.01</td>\n",
"      <td>0.00</td>\n",
"      <td>151.14</td>\n",
"      <td>39.40</td>\n",
"      <td>0.02</td>\n",
"      <td>101.88</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12090</th>\n",
"      <td>25</td>\n",
"      <td>16.64</td>\n",
"      <td>49.97</td>\n",
"      <td>4.05</td>\n",
"      <td>29.26</td>\n",
"      <td>18.80</td>\n",
"      <td>10.03</td>\n",
"      <td>0.52</td>\n",
"      <td>9.84</td>\n",
"      <td>28.30</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>129.27</td>\n",
"      <td>38.14</td>\n",
"      <td>0.00</td>\n",
"      <td>100.80</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12091</th>\n",
"      <td>25</td>\n",
"      <td>15.00</td>\n",
"      <td>66.00</td>\n",
"      <td>0.40</td>\n",
"      <td>26.85</td>\n",
"      <td>14.05</td>\n",
"      <td>5.20</td>\n",
"      <td>0.59</td>\n",
"      <td>2.10</td>\n",
"      <td>17.05</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>128.09</td>\n",
```

```
"      <td>19.15</td>\n",
"      <td>0.00</td>\n",
"      <td>66.24</td>\n",
"      <td>2020</td>\n",
"      <td>7</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>12092 rows × 19 columns</p>\n",
"</div>"
],
"text/plain": [
"       City   PM2.5    PM10      NO     NO2     NOx    NH3      CO    SO2  \\\n",
"0         0  110.71  141.54   63.03  111.56  100.04  26.64   63.03  80.15   \n",
"1         0  147.57  141.54   59.56  107.46  129.87  26.64   59.56  47.70   \n",
"2         0  131.50  141.54  119.68   75.82   88.04  26.64  119.68  55.29   \n",
"3         0  102.12  141.54   57.92   95.29   54.93  26.64   57.92  69.02   \n",
"4         0  115.00  141.54   63.86  111.04   61.99  26.64   63.86  86.65   \n",
"...     ...     ...     ...     ...     ...     ...    ...     ...    ...   \n",
"12087    25   15.02   50.94    7.68   25.06   19.54  12.47    0.47   8.55   \n",
"12088    25   24.38   74.09    3.42   26.06   16.53  11.99    0.52  12.72   \n",
"12089    25   22.91   65.73    3.45   29.53   18.33  10.71    0.48   8.42   \n",
"12090    25   16.64   49.97    4.05   29.26   18.80  10.03    0.52   9.84   \n",
"12091    25   15.00   66.00    0.40   26.85   14.05   5.20    0.59   2.10   \n",
"\n",
"          O3  Benzene  Toluene  Xylene  Vehicular Pollution  \\\n",
"0      57.12     4.08    32.33    6.93               616.55   \n",
"1      48.23     4.10    32.34    6.99               672.20   \n",
"2      43.25     4.09    32.42    7.00               702.90   \n",
"3      51.71     4.09    32.38    6.98               536.36   \n",
"4      59.25     4.12    32.43    6.97               583.93   \n",
"...      ...      ...      ...     ...                  ...   \n",
"12087  23.30     2.24    12.07    0.73               131.18   \n",
"12088  30.14     0.74     2.21    0.38               156.99   \n",
"12089  30.96     0.01     0.01    0.00               151.14   \n",
"12090  28.30     0.00     0.00    0.00               129.27   \n",
"12091  17.05     0.00     0.00    0.00               128.09   \n",
"\n",
"       Industrial Pollution  Organic Pollutants  Inorganic Pollutants  year  \\\n",
"0                    180.61               43.34                501.57  2019   \n",
"1                    139.36               43.43                479.02  2019   \n",
"2                    142.05               43.51                528.40  2019   \n",
"3                    164.18               43.45                413.43  2019   \n",
"4                    189.42               43.52                473.29  2019   \n",
"...                     ...                 ...                   ...   ...   \n",
"12087                 46.89               15.04                 97.07  2020   \n",
"12088                 46.19                3.33                101.38  2020   \n",
"12089                 39.40                0.02                101.88  2020   \n",
"12090                 38.14                0.00                100.80  2020   \n",
"12091                 19.15                0.00                 66.24  2020   \n",
"\n",
"       month  \n",
"0          1  \n",
"1          1  \n",
```

```
"2          1  \n",
"3          1  \n",
"4          1  \n",
"...      ...  \n",
"12087      6  \n",
"12088      6  \n",
"12089      6  \n",
"12090      6  \n",
"12091      7  \n",
"\n",
"[12092 rows x 19 columns]"
]
},
"execution_count": 32,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X_test = final_df.loc[final_df['year'] > 2018][['City', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'Vehicular Pollution', 'Industrial Pollution', 'Organic Pollutants', 'Inorganic Pollutants', 'year', 'month']]\n",
"X_test.reset_index(drop=True, inplace=True)  \n",
"X_test"
]
},
{
"cell_type": "code",
"execution_count": 33,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0        1474.0\n",
"1        1246.0\n",
"2        1719.0\n",
"3        1264.0\n",
"4        1127.0\n",
"          ...  \n",
"12087      41.0\n",
"12088      70.0\n",
"12089      68.0\n",
"12090      54.0\n",
"12091      50.0\n",
"Name: AQI, Length: 12092, dtype: float64"
]
},
"execution_count": 33,
"metadata": {},
"output_type": "execute_result"
}
],
```

```
"source": [
"y_test = final_df.loc[final_df['year'] > 2018].reset_index(drop=True).AQI\n",
"\n",
"y_test"
]
},
{
"cell_type": "code",
"execution_count": 34,
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"c:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\xgboost\\compat.py:36: FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.\n",
"  from pandas import MultiIndex, Int64Index\n",
"c:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\xgboost\\data.py:250: FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.\n",
"  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):\n"
]
},
{
"data": {
"text/html": [
"<style>#sk-container-id-1 {color: black;background-color: white;}#sk-container-id-1 pre{padding: 0;}#sk-container-id-1 div.sk-toggleable {background-color: white;}#sk-container-id-1 label.sk-toggleable__label {cursor: pointer;display: block;width: 100%;margin-bottom: 0;padding: 0.3em;box-sizing: border-box;text-align: center;}#sk-container-id-1 label.sk-toggleable__label-arrow:before {content: \"?\";float: left;margin-right: 0.25em;color: #696969;}#sk-container-id-1 label.sk-toggleable__label-arrow:hover:before {color: black;}#sk-container-id-1 div.sk-estimator:hover label.sk-toggleable__label-arrow:before {color: black;}#sk-container-id-1 div.sk-toggleable__content {max-height: 0;max-width: 0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-1 div.sk-toggleable__content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-color: #f0f8ff;}#sk-container-id-1 input.sk-toggleable__control:checked~div.sk-toggleable__content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-1 input.sk-toggleable__control:checked~label.sk-toggleable__label-arrow:before {content: \"?\";}#sk-container-id-1 div.sk-estimator input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-1 div.sk-label input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-1 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px, 1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position: absolute;width: 1px;}#sk-container-id-1 div.sk-estimator {font-family: monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-1 div.sk-estimator:hover {background-color: #d4ebff;}#sk-container-id-1 div.sk-parallel-item::after {content: \"\";width: 100%;border-bottom: 1px solid
```

gray;flex-grow: 1;}#sk-container-id-1 div.sk-label:hover label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-1 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-1 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-1 div.sk-item {position: relative;z-index: 1;}#sk-container-id-1 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-1 div.sk-item::before, #sk-container-id-1 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-1 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-1 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-1 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-1 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-1 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-1 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-1 div.sk-label-container {text-align: center;}#sk-container-id-1 div.sk-container {/* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 */display: inline-block !important;position: relative;}#sk-container-id-1 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-1\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>XGBRegressor(base_score=0.5,
booster='gbtree', colsample_bylevel=1,\n",
"colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"gamma=0, gpu_id=-1, importance_type=None,\n",
"interaction_constraints='', learning_rate=0.05,
max_delta_step=0,\n",
"max_depth=6, min_child_weight=1, missing=nan,\n",
"monotone_constraints='()', n_estimators=500, n_jobs=10,\n",
"num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0,\n",
"reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact',\n",
"validate_parameters=1, verbosity=None)</pre>In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org. <div class=\"sk-container\" hidden><div class=\"sk-item\"><div class=\"sk-estimator sk-toggleable\"><input class=\"sk-toggleable__control sk-hidden--visually\" id=\"sk-estimator-id-1\" type=\"checkbox\" checked><label for=\"sk-estimator-id-1\" class=\"sk-toggleable__label sk-toggleable__label-arrow\">XGBRegressor</label><div class=\"sk-toggleable__content\"><pre>XGBRegressor(base_score=0.5,
booster='gbtree', colsample_bylevel=1,\n",
"colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"gamma=0, gpu_id=-1, importance_type=None,\n",
"interaction_constraints='', learning_rate=0.05,
max_delta_step=0,\n",
"max_depth=6, min_child_weight=1, missing=nan,\n",

```
"             monotone_constraints=&#x27;()&#x27;, n_estimators=500, n_jobs=10,\n",
"                      num_parallel_tree=1,  predictor=&#x27;auto&#x27;,  random_state=0,
reg_alpha=0,\n",
"                                      reg_lambda=1,   scale_pos_weight=1,   subsample=1,
tree_method=&#x27;exact&#x27;,\n",
"                                                                          validate_parameters=1,
verbosity=None)</pre></div></div></div></div></div>"
],
"text/plain": [
"XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,\n",
"             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"             gamma=0, gpu_id=-1, importance_type=None,\n",
"             interaction_constraints='', learning_rate=0.05, max_delta_step=0,\n",
"             max_depth=6, min_child_weight=1, missing=nan,\n",
"             monotone_constraints='()', n_estimators=500, n_jobs=10,\n",
"             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,\n",
"             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',\n",
"             validate_parameters=1, verbosity=None)"
]
},
"execution_count": 34,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"from xgboost import XGBRegressor\n",
"model = XGBRegressor(n_estimators=500, learning_rate=0.05, n_jobs=10)\n",
"model.fit(X_train, y_train)\n"
]
},
{
"cell_type": "code",
"execution_count": 35,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"0.9778494297545366\n"
]
}
],
"source": [
"score = model.score(X_train, y_train)\n",
"print(score)"
]
},
{
"cell_type": "code",
"execution_count": 36,
"metadata": {},
"outputs": [
```

```
{
"data": {
"text/plain": [
"(12092,)"
]
},
"execution_count": 36,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"y_preds = model.predict(X_test)\n",
"y_preds.shape"
]
},
{
"cell_type": "code",
"execution_count": 37,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"46.49274973537498\n"
]
}
],
"source": [
"from sklearn.metrics import mean_squared_error\n",
"\n",
"rmse = mean_squared_error(y_test, y_preds, squared=False)\n",
"print(rmse)\n"
]
},
{
"cell_type": "code",
"execution_count": 38,
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\426775689.py:4:
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy\n",
"  X_test['AQI_predicted'][ind] = y_preds[ind]\n",
"C:\\Users\\techi\\AppData\\Local\\Temp\\ipykernel_4260\\426775689.py:5:
```

```
SettingWithCopyWarning: \n",
"A value is trying to be set on a copy of a slice from a DataFrame\n",
"\n",
"See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy\n",
"  X_test['AQI'][ind] = y_test[ind]\n"
]
}
],
"source": [
"X_test['AQI_predicted'] = np.nan\n",
"X_test['AQI'] = np.nan\n",
"for ind in X_test.index:\n",
"    X_test['AQI_predicted'][ind] = y_preds[ind]\n",
"    X_test['AQI'][ind] = y_test[ind]"
]
},
{
"cell_type": "code",
"execution_count": 39,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
```

```
"      <th>...</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"      <th>AQI_predicted</th>\n",
"      <th>AQI</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>110.71</td>\n",
"      <td>141.54</td>\n",
"      <td>63.03</td>\n",
"      <td>111.56</td>\n",
"      <td>100.04</td>\n",
"      <td>26.64</td>\n",
"      <td>63.03</td>\n",
"      <td>80.15</td>\n",
"      <td>57.12</td>\n",
"      <td>...</td>\n",
"      <td>32.33</td>\n",
"      <td>6.93</td>\n",
"      <td>616.55</td>\n",
"      <td>180.61</td>\n",
"      <td>43.34</td>\n",
"      <td>501.57</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"      <td>1186.507690</td>\n",
"      <td>1474.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>147.57</td>\n",
"      <td>141.54</td>\n",
"      <td>59.56</td>\n",
"      <td>107.46</td>\n",
"      <td>129.87</td>\n",
"      <td>26.64</td>\n",
"      <td>59.56</td>\n",
"      <td>47.70</td>\n",
"      <td>48.23</td>\n",
"      <td>...</td>\n",
"      <td>32.34</td>\n",
"      <td>6.99</td>\n",
"      <td>672.20</td>\n",
```

```
"      <td>139.36</td>\n",
"      <td>43.43</td>\n",
"      <td>479.02</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"      <td>1009.924927</td>\n",
"      <td>1246.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
"      <td>131.50</td>\n",
"      <td>141.54</td>\n",
"      <td>119.68</td>\n",
"      <td>75.82</td>\n",
"      <td>88.04</td>\n",
"      <td>26.64</td>\n",
"      <td>119.68</td>\n",
"      <td>55.29</td>\n",
"      <td>43.25</td>\n",
"      <td>...</td>\n",
"      <td>32.42</td>\n",
"      <td>7.00</td>\n",
"      <td>702.90</td>\n",
"      <td>142.05</td>\n",
"      <td>43.51</td>\n",
"      <td>528.40</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"      <td>1558.571655</td>\n",
"      <td>1719.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0</td>\n",
"      <td>102.12</td>\n",
"      <td>141.54</td>\n",
"      <td>57.92</td>\n",
"      <td>95.29</td>\n",
"      <td>54.93</td>\n",
"      <td>26.64</td>\n",
"      <td>57.92</td>\n",
"      <td>69.02</td>\n",
"      <td>51.71</td>\n",
"      <td>...</td>\n",
"      <td>32.38</td>\n",
"      <td>6.98</td>\n",
"      <td>536.36</td>\n",
"      <td>164.18</td>\n",
"      <td>43.45</td>\n",
"      <td>413.43</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"      <td>1078.675049</td>\n",
```

```
"      <td>1264.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0</td>\n",
"      <td>115.00</td>\n",
"      <td>141.54</td>\n",
"      <td>63.86</td>\n",
"      <td>111.04</td>\n",
"      <td>61.99</td>\n",
"      <td>26.64</td>\n",
"      <td>63.86</td>\n",
"      <td>86.65</td>\n",
"      <td>59.25</td>\n",
"      <td>...</td>\n",
"      <td>32.43</td>\n",
"      <td>6.97</td>\n",
"      <td>583.93</td>\n",
"      <td>189.42</td>\n",
"      <td>43.52</td>\n",
"      <td>473.29</td>\n",
"      <td>2019</td>\n",
"      <td>1</td>\n",
"      <td>946.966309</td>\n",
"      <td>1127.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>195</th>\n",
"      <td>0</td>\n",
"      <td>42.43</td>\n",
```

```
"      <td>116.70</td>\n",
"      <td>32.10</td>\n",
"      <td>114.44</td>\n",
"      <td>92.42</td>\n",
"      <td>26.64</td>\n",
"      <td>32.10</td>\n",
"      <td>142.27</td>\n",
"      <td>38.20</td>\n",
"      <td>...</td>\n",
"      <td>80.04</td>\n",
"      <td>1.48</td>\n",
"      <td>456.83</td>\n",
"      <td>268.29</td>\n",
"      <td>87.82</td>\n",
"      <td>478.17</td>\n",
"      <td>2019</td>\n",
"      <td>7</td>\n",
"      <td>558.267273</td>\n",
"      <td>455.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>196</th>\n",
"      <td>0</td>\n",
"      <td>38.99</td>\n",
"      <td>114.94</td>\n",
"      <td>31.97</td>\n",
"      <td>103.79</td>\n",
"      <td>87.09</td>\n",
"      <td>26.64</td>\n",
"      <td>31.97</td>\n",
"      <td>156.48</td>\n",
"      <td>32.57</td>\n",
"      <td>...</td>\n",
"      <td>80.04</td>\n",
"      <td>1.48</td>\n",
"      <td>435.39</td>\n",
"      <td>276.87</td>\n",
"      <td>87.82</td>\n",
"      <td>470.51</td>\n",
"      <td>2019</td>\n",
"      <td>7</td>\n",
"      <td>559.851257</td>\n",
"      <td>488.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>197</th>\n",
"      <td>0</td>\n",
"      <td>34.92</td>\n",
"      <td>114.11</td>\n",
"      <td>30.50</td>\n",
"      <td>110.39</td>\n",
"      <td>88.63</td>\n",
"      <td>26.64</td>\n",
"      <td>30.50</td>\n",
```

```
"      <td>116.97</td>\n",
"      <td>37.07</td>\n",
"      <td>...</td>\n",
"      <td>80.04</td>\n",
"      <td>1.48</td>\n",
"      <td>435.69</td>\n",
"      <td>241.86</td>\n",
"      <td>87.82</td>\n",
"      <td>440.70</td>\n",
"      <td>2019</td>\n",
"      <td>7</td>\n",
"      <td>600.163086</td>\n",
"      <td>513.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>198</th>\n",
"      <td>0</td>\n",
"      <td>32.70</td>\n",
"      <td>105.78</td>\n",
"      <td>26.00</td>\n",
"      <td>103.72</td>\n",
"      <td>80.25</td>\n",
"      <td>26.64</td>\n",
"      <td>26.00</td>\n",
"      <td>132.41</td>\n",
"      <td>45.92</td>\n",
"      <td>...</td>\n",
"      <td>80.04</td>\n",
"      <td>1.48</td>\n",
"      <td>401.09</td>\n",
"      <td>266.15</td>\n",
"      <td>87.82</td>\n",
"      <td>440.94</td>\n",
"      <td>2019</td>\n",
"      <td>7</td>\n",
"      <td>542.183105</td>\n",
"      <td>466.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>199</th>\n",
"      <td>0</td>\n",
"      <td>44.65</td>\n",
"      <td>120.74</td>\n",
"      <td>25.34</td>\n",
"      <td>135.12</td>\n",
"      <td>94.80</td>\n",
"      <td>26.64</td>\n",
"      <td>25.34</td>\n",
"      <td>104.15</td>\n",
"      <td>58.38</td>\n",
"      <td>...</td>\n",
"      <td>80.04</td>\n",
"      <td>1.48</td>\n",
"      <td>472.63</td>\n",
```

```
"      <td>250.35</td>\n",
"      <td>87.82</td>\n",
"      <td>469.77</td>\n",
"      <td>2019</td>\n",
"      <td>7</td>\n",
"      <td>493.770660</td>\n",
"      <td>429.0</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>200 rows × 21 columns</p>\n",
"</div>"
],
"text/plain": [
"     City   PM2.5    PM10      NO     NO2     NOx    NH3      CO     SO2  \\\n",
"0       0  110.71  141.54   63.03  111.56  100.04  26.64   63.03   80.15   \n",
"1       0  147.57  141.54   59.56  107.46  129.87  26.64   59.56   47.70   \n",
"2       0  131.50  141.54  119.68   75.82   88.04  26.64  119.68   55.29   \n",
"3       0  102.12  141.54   57.92   95.29   54.93  26.64   57.92   69.02   \n",
"4       0  115.00  141.54   63.86  111.04   61.99  26.64   63.86   86.65   \n",
"..    ...     ...     ...     ...     ...     ...    ...     ...     ...   \n",
"195     0   42.43  116.70   32.10  114.44   92.42  26.64   32.10  142.27   \n",
"196     0   38.99  114.94   31.97  103.79   87.09  26.64   31.97  156.48   \n",
"197     0   34.92  114.11   30.50  110.39   88.63  26.64   30.50  116.97   \n",
"198     0   32.70  105.78   26.00  103.72   80.25  26.64   26.00  132.41   \n",
"199     0   44.65  120.74   25.34  135.12   94.80  26.64   25.34  104.15   \n",
"\n",
"        O3  ...  Toluene  Xylene  Vehicular Pollution  Industrial Pollution  \\\n",
"0    57.12  ...    32.33    6.93               616.55                180.61   \n",
"1    48.23  ...    32.34    6.99               672.20                139.36   \n",
"2    43.25  ...    32.42    7.00               702.90                142.05   \n",
"3    51.71  ...    32.38    6.98               536.36                164.18   \n",
"4    59.25  ...    32.43    6.97               583.93                189.42   \n",
"..     ...  ...      ...     ...                  ...                   ...   \n",
"195  38.20  ...    80.04    1.48               456.83                268.29   \n",
"196  32.57  ...    80.04    1.48               435.39                276.87   \n",
"197  37.07  ...    80.04    1.48               435.69                241.86   \n",
"198  45.92  ...    80.04    1.48               401.09                266.15   \n",
"199  58.38  ...    80.04    1.48               472.63                250.35   \n",
"\n",
"     Organic Pollutants  Inorganic Pollutants  year  month  AQI_predicted  \\\n",
"0                 43.34                501.57  2019      1    1186.507690   \n",
"1                 43.43                479.02  2019      1    1009.924927   \n",
"2                 43.51                528.40  2019      1    1558.571655   \n",
"3                 43.45                413.43  2019      1    1078.675049   \n",
"4                 43.52                473.29  2019      1     946.966309   \n",
"..                  ...                   ...   ...    ...            ...   \n",
"195               87.82                478.17  2019      7     558.267273   \n",
"196               87.82                470.51  2019      7     559.851257   \n",
"197               87.82                440.70  2019      7     600.163086   \n",
"198               87.82                440.94  2019      7     542.183105   \n",
"199               87.82                469.77  2019      7     493.770660   \n",
"\n",
"        AQI  \n",
```

```
"0    1474.0  \n",
"1    1246.0  \n",
"2    1719.0  \n",
"3    1264.0  \n",
"4    1127.0  \n",
"..      ...  \n",
"195   455.0  \n",
"196   488.0  \n",
"197   513.0  \n",
"198   466.0  \n",
"199   429.0  \n",
"\n",
"[200 rows x 21 columns]"
]
},
"execution_count": 39,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X_test.head(200)"
]
},
{
"cell_type": "code",
"execution_count": 40,
"metadata": {},
"outputs": [],
"source": [
"city_median_AQI_per_year_test                                                                    =
X_test[['City','AQI','year']].groupby(['City','year']).median().reset_index()"
]
},
{
"cell_type": "code",
"execution_count": 41,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
```

```
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>year</th>\n",
"      <th>AQI</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2019</td>\n",
"      <td>455.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2020</td>\n",
"      <td>146.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Aizawl</td>\n",
"      <td>2020</td>\n",
"      <td>23.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Amaravati</td>\n",
"      <td>2019</td>\n",
"      <td>76.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Amaravati</td>\n",
"      <td>2020</td>\n",
"      <td>54.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5</th>\n",
"      <td>Amritsar</td>\n",
"      <td>2019</td>\n",
"      <td>94.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
"      <td>Amritsar</td>\n",
"      <td>2020</td>\n",
"      <td>81.000000</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>7</th>\n",
"      <td>Bengaluru</td>\n",
"      <td>2019</td>\n",
"      <td>89.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>8</th>\n",
"      <td>Bengaluru</td>\n",
"      <td>2020</td>\n",
"      <td>77.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>9</th>\n",
"      <td>Bhopal</td>\n",
"      <td>2019</td>\n",
"      <td>146.500000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>10</th>\n",
"      <td>Bhopal</td>\n",
"      <td>2020</td>\n",
"      <td>111.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>11</th>\n",
"      <td>Brajrajnagar</td>\n",
"      <td>2019</td>\n",
"      <td>116.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12</th>\n",
"      <td>Brajrajnagar</td>\n",
"      <td>2020</td>\n",
"      <td>126.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>13</th>\n",
"      <td>Chandigarh</td>\n",
"      <td>2019</td>\n",
"      <td>107.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>14</th>\n",
"      <td>Chandigarh</td>\n",
"      <td>2020</td>\n",
"      <td>71.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>15</th>\n",
"      <td>Chennai</td>\n",
"      <td>2019</td>\n",
"      <td>92.000000</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>16</th>\n",
"      <td>Chennai</td>\n",
"      <td>2020</td>\n",
"      <td>76.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>17</th>\n",
"      <td>Coimbatore</td>\n",
"      <td>2019</td>\n",
"      <td>77.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>18</th>\n",
"      <td>Coimbatore</td>\n",
"      <td>2020</td>\n",
"      <td>74.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>19</th>\n",
"      <td>Delhi</td>\n",
"      <td>2019</td>\n",
"      <td>209.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>20</th>\n",
"      <td>Delhi</td>\n",
"      <td>2020</td>\n",
"      <td>148.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>21</th>\n",
"      <td>Ernakulam</td>\n",
"      <td>2020</td>\n",
"      <td>95.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>22</th>\n",
"      <td>Gurugram</td>\n",
"      <td>2019</td>\n",
"      <td>173.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>23</th>\n",
"      <td>Gurugram</td>\n",
"      <td>2020</td>\n",
"      <td>141.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>24</th>\n",
"      <td>Guwahati</td>\n",
"      <td>2019</td>\n",
"      <td>91.000000</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>25</th>\n",
"      <td>Guwahati</td>\n",
"      <td>2020</td>\n",
"      <td>148.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>26</th>\n",
"      <td>Hyderabad</td>\n",
"      <td>2019</td>\n",
"      <td>94.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>27</th>\n",
"      <td>Hyderabad</td>\n",
"      <td>2020</td>\n",
"      <td>73.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>28</th>\n",
"      <td>Jaipur</td>\n",
"      <td>2019</td>\n",
"      <td>110.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29</th>\n",
"      <td>Jaipur</td>\n",
"      <td>2020</td>\n",
"      <td>106.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>30</th>\n",
"      <td>Jorapokhar</td>\n",
"      <td>2019</td>\n",
"      <td>114.651163</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>31</th>\n",
"      <td>Jorapokhar</td>\n",
"      <td>2020</td>\n",
"      <td>138.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>32</th>\n",
"      <td>Kochi</td>\n",
"      <td>2020</td>\n",
"      <td>98.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>33</th>\n",
"      <td>Kolkata</td>\n",
"      <td>2019</td>\n",
"      <td>97.000000</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>34</th>\n",
"      <td>Kolkata</td>\n",
"      <td>2020</td>\n",
"      <td>92.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>35</th>\n",
"      <td>Lucknow</td>\n",
"      <td>2019</td>\n",
"      <td>193.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>36</th>\n",
"      <td>Lucknow</td>\n",
"      <td>2020</td>\n",
"      <td>136.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>37</th>\n",
"      <td>Mumbai</td>\n",
"      <td>2019</td>\n",
"      <td>93.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>38</th>\n",
"      <td>Mumbai</td>\n",
"      <td>2020</td>\n",
"      <td>82.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>39</th>\n",
"      <td>Patna</td>\n",
"      <td>2019</td>\n",
"      <td>184.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>40</th>\n",
"      <td>Patna</td>\n",
"      <td>2020</td>\n",
"      <td>147.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>41</th>\n",
"      <td>Shillong</td>\n",
"      <td>2019</td>\n",
"      <td>44.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>42</th>\n",
"      <td>Shillong</td>\n",
"      <td>2020</td>\n",
"      <td>49.883721</td>\n",
"    </tr>\n",
```

```
"    <tr>\n",
"      <th>43</th>\n",
"      <td>Talcher</td>\n",
"      <td>2019</td>\n",
"      <td>123.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>44</th>\n",
"      <td>Talcher</td>\n",
"      <td>2020</td>\n",
"      <td>117.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>45</th>\n",
"      <td>Thiruvananthapuram</td>\n",
"      <td>2019</td>\n",
"      <td>69.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>46</th>\n",
"      <td>Thiruvananthapuram</td>\n",
"      <td>2020</td>\n",
"      <td>64.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>47</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>2019</td>\n",
"      <td>106.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>2020</td>\n",
"      <td>81.000000</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"                  City  year         AQI\n",
"0            Ahmedabad  2019  455.000000\n",
"1            Ahmedabad  2020  146.000000\n",
"2               Aizawl  2020   23.000000\n",
"3            Amaravati  2019   76.000000\n",
"4            Amaravati  2020   54.000000\n",
"5             Amritsar  2019   94.000000\n",
"6             Amritsar  2020   81.000000\n",
"7            Bengaluru  2019   89.000000\n",
"8            Bengaluru  2020   77.000000\n",
"9               Bhopal  2019  146.500000\n",
"10              Bhopal  2020  111.000000\n",
"11       Brajrajnagar  2019  116.000000\n",
```

```
"12        Brajrajnagar  2020  126.000000\n",
"13          Chandigarh  2019  107.000000\n",
"14          Chandigarh  2020   71.000000\n",
"15             Chennai  2019   92.000000\n",
"16             Chennai  2020   76.000000\n",
"17          Coimbatore  2019   77.000000\n",
"18          Coimbatore  2020   74.000000\n",
"19               Delhi  2019  209.000000\n",
"20               Delhi  2020  148.000000\n",
"21           Ernakulam  2020   95.000000\n",
"22            Gurugram  2019  173.000000\n",
"23            Gurugram  2020  141.000000\n",
"24            Guwahati  2019   91.000000\n",
"25            Guwahati  2020  148.000000\n",
"26           Hyderabad  2019   94.000000\n",
"27           Hyderabad  2020   73.000000\n",
"28              Jaipur  2019  110.000000\n",
"29              Jaipur  2020  106.000000\n",
"30          Jorapokhar  2019  114.651163\n",
"31          Jorapokhar  2020  138.000000\n",
"32               Kochi  2020   98.000000\n",
"33             Kolkata  2019   97.000000\n",
"34             Kolkata  2020   92.000000\n",
"35             Lucknow  2019  193.000000\n",
"36             Lucknow  2020  136.000000\n",
"37              Mumbai  2019   93.000000\n",
"38              Mumbai  2020   82.000000\n",
"39               Patna  2019  184.000000\n",
"40               Patna  2020  147.000000\n",
"41            Shillong  2019   44.000000\n",
"42            Shillong  2020   49.883721\n",
"43             Talcher  2019  123.000000\n",
"44             Talcher  2020  117.000000\n",
"45  Thiruvananthapuram  2019   69.000000\n",
"46  Thiruvananthapuram  2020   64.000000\n",
"47       Visakhapatnam  2019  106.000000\n",
"48       Visakhapatnam  2020   81.000000"
]
},
"execution_count": 41,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"city_median_AQI_per_year_test['City']=le.inverse_transform(city_median_AQI_per_year_tes
t['City'])\n",
"city_median_AQI_per_year_test"
]
},
{
"cell_type": "code",
"execution_count": 42,
"metadata": {},
```

"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABTcAAANaCAYAAABcDngdAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjYuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8o6BhiAAAACXBIWXMAAA9hAAAPYQGoP6dpAADGiklEQVR
4nOzdd5hU5f0/7vfSe2+iICiIIBbEhg1QwV5jNGIUsCuC2DV+EUtEXURscQGEsQusSVIEVBUIqAo9oYCkRUUpHf
O7w9/zMd1l6YLMwfv+7rmutwzz555zTjszLzmOc/JS5IkCQAAAACAlCmR7QAAAAAAAAAAL+CopNwEAAACAVFJuAA
QSspNAAAAACCVlJsAAAAAAQCopNwEAAACAVFJuAgAAAACppNwEAAAAAAFKpVLYDAAAAAEDaJUkSK1eujFWrVrmVmU7Suq
VLFkySpUqFXl5eesdq9wEAAAAgN9g+fLlMXPzmzFi8eKL8hSKlSoEFtttVWUKVNmnePykiRJNlMmAAAAAANiirF6
9Oj7//PPMoWbWjJk1K5dOS8c0qUKbNBMw4pWPIksXz58pg9e9e3asWrUqmjZtZtTGiVKrH1lTTM3AQAAAOBXWr58
eaxevToaNGsGFSpUyUUyHacLUL58uWjdnOnS8c0338Ty5cujXLlyax3hbEIAAAAAAAAAAAAA8Buta3s5+yS7iAAAAAA
AAAAAAoAqKAQCQFSpNAAAAAAECCVlJsAAAAAAAhABBpvgUAAAAAOAApEjAAAAQi0ASpwoYBIGXgtOWrFWriUmUDIfHjf
3jSNGBMIcgDIdRBaAMAAcPFyvHmoYGkqHCMQGNBSNQT8Y6/D4cADYpLXxmWfz4uAAEHCXg+lYXpBkjKJzBhCKIRyV
j2ehxMeoZGUx2/PUUgBGJsV1+Tggh8RQTuVsDvQmyjQEeEYHFq1Ax7ZCs8Kid4QAAAYYZFmOD+ez8uoOAZScAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
4ceMoV65cVKpUKXbffffo27dvzJkzJzOuXbt20a5fGl9nTB0nYY5Lg7bdX8yJqPXM/k8sZ8GwqEIb/1DDn1TiEaKoN1wnQ8y
QkSzV7GHkSAv8Hp8/Ou84+l+O+s8c23PeVmcw7/Gub9DwL7xCAy8U5QiVudVEXXH9QkoCi/Mx6qsmnq5V+H1Yq/ozw1e
VDzeWPPqp8rfbZLjCwIvGLDP/E8lGAWdGBx9EQZi6c4x4jgOnLPmCzPLs6aVnXN2tlSq+vSVXIqJ7PAyGmXA2ok06MJAGh
K8nTVJ0onCn72E8ZrLx6AZIrGF2gT6YG2zSrwb+/Pv+2+v8t/YLt+TDh7xY/N0Zjkmp/qdR/Nu8JPs1ZSMd+mnfeCeh
kJmFOXT9d0l7Mt/JMpOlxDlTDx6GPb1d+ZhB3uurPm/VuY2uupPFOu+KudiKsbl6k+rz8X7hFA5Qm/W/xx0XBcn9ie9Wkk
SG1RY7XkZ37YsE8VvLBk3hu0nv8sYZQDDwwC3g1xUOtjsAfn5VH/d6i1JHjF9wvo+FMVEgCjDFBWHfM1RunBBK3g
Ozy+iqN3X3cWhkIGJ7LrALwfAZ1qFz9NKD0AbHjmhk37gjBLPE6PWlk0ddbOsgJoP9XOGd/Fh1/FjoP9TBeeOXwNoYKYK/
x6NGh0VOFfE3xhT8Rf5AvXqLUOk+6eKwNjpf9rFq/IGbZtrtKiDgNGJXbyVAGfy6F9WF9efnQvddn0ld58xsNKgswPmh
MnQNS0hsO7PDCNHnHMyHHRiYQN7LGd3n9AR4ysVJIXoSnOmI+BnxnmaMx5m6gSa1+iK2ZqDs7+Y4EmzDV7KK1I88JYo46J
cO42KJyl6nKe3pK+SZ2KWsFTKHnaHHkG5lHxpBbKAg3X0B7SQhbb8qO9c6DDAeCw3wWUMPqg8B9W9YT4yx0FS07m+/5
dDdtC1fd9n4QCCJ+5m+35NN4JZ7OFy7rbpdeM4ZXKxeu+zvvs4p+GsJOrcJdbm7DL4y1TOF1TmLK2Z6kCLdwNrh/bnbQL
zYAdHVVyRdW8O2cz8JHl1JBMOuOsmtx2vvNBA+xlMUwm4Y+/p/xPkQcx/TJjrbx4vBk4WGmNENixm2SJiv+XUrY5Wdp
33ZWHx/SaeCIUHzEJYuTzlGpzPHEj7PDmk8ncsTFRtKxJz/mQmCt+q6aT3EJ0/nYIsH/GZ3q4G7vYF9Ld2N3mRMRn4rZ
FExb7FWsHIsxY/2RAtnJmxl+GTUNLcPjp7o4pM58jIEIuuPB5Q8UrxN9Uf12ezh8Qwl0LK2CH/BmQWP1KlfMaofNU56b
G6Q75PDgjm7X3l1oNlCEjqU+ED2U+1+0wRj0a4a/MMuQ1uvEqdvpdMJlIc0VjKPPMsE3UQBwbXMD63TjqUkTZQPLFgSB
5QDzpFBxbvBO2gNKaTNfEVV4TXjWDc0XjqXV1WT5w1nFiEPlLRXR0SKCa1ErY3aH8DDkT0BItUO1nMwRD5MYkBD6Kbe
7iPTUy8bZ9D4D9DZf1HB9XjpyJU+E/5l0JeTQ1FjuYYERmFcEbvPjGFzaMsa8UTD+1KyhHOuZQHwGQXy4xmJ5pszG8JQ
3ByVyY5yAjhMIJ8fHAkjpK0G8YEg7IZXkpJtVKABBJldV06Tlk3uwcEAUD6AHuL+HbA4XjIbe/F12yDkI9VAfjVcVeRj
UXOY2SdwP4lOjX8mEEXgcodgbXbl5FLi2pgl2BIjk6svF1vKuWb71wi9THGfsX2BEQpWLGR75bqdEx0U4ziR1QSXR9FY
HDgmW6B7xJp8rhEVsvqOnr6SdV0wV+QLZyUnGsjGuVWQTyfZ2GnhL7ngPQ1xvsjPzAw8Otl+AIMxFGm6rD6bHqz3FKMXOyU
ukGgFG5hwoD3mQ9q5pvCnB/mP6Vab1rOR3eyo0DItmdwI/1ve2tnG5e+GVVZ5jbMvVM3vuK8G/tvrw6nM+X9ngtOQzj
g0m0YjWjXM/EVFB0rV8HNjv6B8tgjgYu79c19qM5G2CdDH/kwxTUY/B2PvqZAgAw/Yu76fq9Yz9uI/tzfuvdVV28vw0
F+wF/DULZ8U2a2WFd+JFDM9lD9jTWBdlHLJyXMNV5fm94Qo5zu9+Txo5jzz8mKzebbQfCcBBVZNyeNwY8ZHW/+4bHIJ
1jz/IuX6LFtmGJySYtyy/LqbcTMB5AiVbnn7Lwgd5Ukq8N8nzhuVrt5X3pqd5ZWxqdfbc7HPP62b8NxnMsbefzyHfGy
/9gV+rsZa/ZK0abD0RMXbngmKxjw0ZjS/4bRhZ9rprMXlVGS1TpkNdQNdbSW0DGJDF8QvdwCvYMHl3dY5kwFQrVYhAB
Gcuh9JxMkYIBPMm36Vv1y+4MOk9IrJqHeNGpIXZX4TS37b2mzOvQy1OQPX+SaFpH7xBuX1ZvyuUtEuLvvC/mNE0arDJq
s2LCIiDjzzDPjrbfeirfeeit69eoVBxxwQ6RMR0b9//9hzzz1/9zzV+KzzXqp3DDwnnd+MK3RyNMytT+y6Vm7cXpj
fN2/PEaNtPUCj9DxXDNNmiCbQb1GxNCePqHrPyR7rBo9ShNT84CXN6L2kiN/FKnkAhpiGHr3pfP0EHS7xYpxJeypo57
GnTp14ogjioiINNO20N9y6ekSqZU3hQtrqKpSZz2i/cvb9ZHkzB+AcGYAkAPxBEWJHsAmj6KJU2rD3dx2f8Cxt0zHlwDIt
o5ZJ2ykXUF3ncVMCvW8prANdvwY8xZiSkRLUXzSBOP9Q8NbhjSLtXCBkMd64wa3jaDJr8wFv/G6cf0YVzR09zWDiEB0r
KE3zvF5Hj80MOvOjoCbOHYNuM1/gDKLHX9Uu4FHxgkvBTPfAVcDtbFN5+CBHm4rArDb6ZBE8t64aHMeL1OXkQ3G0hoZa
88smFxq9evTqOPfbYuOWW6bHz8WTp8JTp07x8ssy8233BIjRoyIdu3azaFHttt/bZZ8ff/va3oP300+P555+PP/zhD3Hg
E3LLlz5vr66+/jmbNmkW/VdeiVtvvTVmzpwVbcnkWe65Z3z//feZZpvP9PVVr61iQT9gX7F9K7LovOjM+VavmQmMlmPR
1xnJKMC7JkyrCFfCt4UJBB8Vl5h5UmUg4Bq8BmnELCKKcBSbNNRkHyiS+d9KXU06OWzAx5n9Mli91VmFwOIRw3cO+x/Z
OfzQ2Iw7n9+yENwdQHfBLjAR65NS1Hui47QVMnpZhNbdM+XbemNWf0+a9U3x/JtQ+q3Dr+x/iO6k+jh+kz4kbMFdj/9D
e8dXpeoZx1rtVD2zDdSYU3x0eTdHuTS4ocBrIKl1RMi0R0/dTwxHQeNCb1u7ZtHOCOnfvZ5Z4d7C7bnwxOkvc+fGIEbg
BA9Cbz+3NptxgWeV2l9XjgdVdE0FajFOEbZFiXcP0lOVmudiaWyq2R4T8nrdAyVVAFx5zXPlv1MXVUtuIkZBj3NrX8d
5DD2Yfv4RX0r0zXgrsi6iDw7Hb9o1cWLbkXjOXfPe5H6KJVh1X1+W6bLvr8ZnZUGsVrmX2H3SrmcN33AmHD3zsGNdi
KNiWapRDrTMFv0gEv5H4lIKPMd2ILUk/TimzSQw2FOUYBTRlx9f2vnbTAsD+tGm57/WKoHTbEfKqbh0MNhf/2K70kYk+
pBdQuKW6RkL2zvWPdIcLYV7uiDOcyKZ+kDB2YJgy7HWzi9p2MuHT0lvZ+Ta0K2UX6CwvC+Kpiv7lOYUzyaF5Tbb8yZ9q
YNT0XHCsI5/rDsX/aQE8GKlWmO3yeJFkhMQJvpSntpfPexDGrRsWT/L06KsNXyDFe0xY3/jB6NW1jF4iPM23edaK4h6
mDSJ4pOfFSMhX6lTEiDmRuomOOu2HJJ0E/D87r4mn4GRHoz9JlH61qqwHZzVZq4qXHTIwAnw6TtfAdsoXd2XUQzybL
+75CHZn0zAfVyw1JapmfMJtNJCUTDsF8LyrWvvE+bS7W2PNOjz1mzxkCtzU5WGMMnbhVzSSbwqfhEl1IoLjSYOTmY5
wBo7nJXibRXb4UM0i6Ckp7f3bFVlbfGyxqd4QDBoCJ7DwXvX+cz1JDE1pnGCsAQ9k1P6jYQ0Rt7yJHlfXUs0Wu3u2Vb
TLJlqLvFvHZrXxuSqdftXlBKrOLBBq8Rq0+6a3gl8dqvUm6DKOPV/RbCkTJCLczvdv4t6IzmE6t4QlfVaWx8zePQvTY
MgFVyRCh/Ptd3Pr+pf1fLoNvjx9lvpDNeW9NqJQ74pDeEtFDBv2z3LHO0RMh4s2wGdPQMxBhANfmyWwOLjxEWHgiPLDzb
rpV6RBSqbxk1ArT4YXgOC9MhFlQ6TG/3x0HDxr0UTMpSbC6FU6o+yvmHtsZewYr6DUyBEywqFwbXTRQ5zG8hLLVcMFZ
LVVK1qXp8ZScpkEkZaLlvR5tBD9V4H/EC3tULzdIczg4KY7IZs7XDOxrBVPtRTfSnLsudsrQvfQ2pJHBv+kFTzEQvy9I
IeTxcsHhY6ey8v+O3zPj8btSQxOEY/ayYWO5pyq5t0H1fYnLbhRFFS2mltZZRr+BpCn7o9nz0+u5JfdInNwm6ZoZbuf8
FGvJcfUNNr4tmrbd7nbTuTfddUQuVyTX/RvhzIwCmfuZCa1sn7iTH+K6n4IKm8s+q5lYWhpu6xtgK/rl3WBjvvmPTb0
lb3VNTWfm5lSGGIm8WmMu4qLN2b9HLCq+pEwk1X4OUfMpjy7rhNlXGqYs0lBtqF8mnBYhAmNq7sjLJ9ecYzRqG8u8Ezp0N
ctjUmGWHg5sj1ge6d8YrNE07iUGR5CjhdMutWfGuG/UdupaJ/Rd34cCddeOXGHqK3NIo4QgVKxt97rfIu14IVp5ivaYl
2hmyR1uQaPH1kGMQE6aNdxnCBlCo6fHodevjhSdwm0QqmyeOqV3WsPK8H/ftY+7i0SGT1kV4LS8ZvP3fQZ/nxYBSSKHz
O5Zd+EeK6CUIz0A/9mhhCKdt2yLN0NW4K7ecc3qLSC5WYOGUx0NJlnHqoNtXc7ekkXPwCvdc2C4Ra6/hfKP4x5YNJ7O3x
HGDSJe8yyiTUXHcyiD1fLRs1fvMYzi/5pp9tJEmQrfSFYPDrkMWLIp/2t4eawd6EQoNtwtcXZOk6SFDG8gNjhFsZMKSB
8/aCqQ+6Xz3ch76hNN6gugY3OuHvMjUfmFfqkeMl8fQ7sA79LmAT9qO34jwM2HtvfnOEsgrCc+yHvbFc4o9zlX8BhFVwd8
5UtgIQ9vNuF4Yn1Wx3vdxqR1tpPcDVaFEbOlStPJd9w5ewO6t7DW6yjqjzOKD2IPmchJkaKkotx0O02H7RdqN2yNOpW
E4+0cLq3i01xa6q9ivyeoH8AZ9g+pj9PGEP58yxoUlsFLzfsgKPP1BeFafrhV71RJm3OFv5/4XBHKmv0/pOv
jjsDyoJe0aKk6IL2j+Pp/6zCPZjIojMh0ejm1Lsf9yQvCzY38vTX8Iq+/7vrsf0MnWrjzTaFQ9kgjTMSOzEeJ5HsuUvyWRvt2f
8t1s8srR+q3FRyjvU9gBRWEVn96gTTXb07t/CBtj9QZQlhK85/2Yc+Tg03fcfw8qkc8x1c1IhB6CcsKjDUGP8jFocrO/x
fZg62FaXyNgpyg0lfIsMxN6dBB/3jM2T/lVqcY2dYcS8yBkLB9R3v72edvNYoZf2adh0xtpAXYJBcQ35wW7x79yB1c
e4B6/h3dq6/3hqqnYVdLnz/2x6P7PYdnkn7WcFLQdVq/QNp5pFqFeTaL3KRmpRFhZRgVzzRqhSlSQIQ4aWr6i+xfZ5oq69v7
TxPL1r1fRzpw2tEWzmsbJ4Jpk0kd9njm/uWXx5UJh9TqD8nvjdYMVZ/DKFr+0+1R9kc1tvMXT6bnaqgw9OPr4D+Gp0Ad
iLp5nbq0Hk6Gbig/HLEUBxX4y3qa2H/oxmZZoJb0Lv1XjR1eYEXUi3//cqdy/RmJJ/Xzz8mY9x8uG+N21xAlWYiKySy
tEeLtGT3D7uHkGLaGnPwN9ceWW/YXmiR37f0lO8g/d+KdBqGlpTf/2VxJrOcf6bbMyA/ohtsbTd7fTthjM2q6n6LpKZ
UiZlwEh9a3xUUOYNvXk6E0+79ZaFmjxS3wd+Zz9CPLsU2J3L/6ypabbnVc8qm2MDxJ4i+6BUgrjNC9PYT0G+QxvlPFRe
VLCTqkQSxNB56cSQtTwXxcFFbcOnwhTi7tDLbdG2l4pBlfzt6WNHpfIS6f+/f4Y3q5RdGelPIr4l4Wi5J/nt/+TdDHdB0oab
CyDcGWSg2BJ2mQvgvtLlJ57iVI6Zh3lijBmoxCk2sQtf/Pn0nSyeKu5sfl7Cs1iZ2uAXb6O1TdnyzUdn97swypNW+V1D
dNmUl/Mtk18oFUdeLG9jXwM0ybSN3J3JVk87X9W0YMZBkbhybpaEpUI5k0jA9Ng/TTSHR2tE6ymA9GH23C7gMSdtwx9F
4dNZJRO7wuqCOn6NJIHbNWxVDvmQHyZ0PoCmKe22oyVZbjMMs4zXh6uUHlmMsg5R4TpOaAr6NnUmmY9E1cJFTijwpa
aOCC1OsTh4R+1W8DsgICdh6ltqYAQwH+KcvZrDmF6K0EhL6JwYwJcdm7bZy8YHCaXjEYbXg6uG/xQChjdH5sQLpmwCqbh
pIhvRn7DR6mCa7G0y3hXVjoGRaIGEn3jx35mCNG3h0GYZ71B3ErIoJ1Y6bi5c3bGwYKiyqZ8dcfVa4t2rN7pFfrNMnF/WM
Dg4wHX1ClLLKAcb92Uz2YaLJZFMhv3ZlawEPQrtRcRJBjO0wgcgNDQ6oTlkCpTQs2Kl5Z5Ctg9kdthNu6dnGL2YyYzL6/
W7nGnbWPcvRbDOyHH4SIZJPE7xX6yFR5yCVE2Y42GgEF4iEHYhCNiBR83lPKJcIrpf2EBw/lY0X8eDbl5mqqf2YTKpIOGlrL
4dE6DD/2nHSL5OXWQEFcUE8kq9NOLWUR9LoFbf2yqfiVdrr25HqrjcV8AkvpVmMaOGYx0bjq8PQF5yRmhU1N2FM4vOGf
yTFV28TdR0KA3FhDLwFXQeKUjjnbZpvVqFCGcBHFzgo3YjKgThDtPYvyzmbCc9oJ8sL/IFs+IXXq6FWkbhAw8Lrc4bd6
2H8YNvWOI4J+vXhNo4BbX67+yRGCNwHcV6jsAsBvtJljT6XLdxNGAcS3Eu3WLeIMlWp8WbdOaWlTtpXdbd5O4g+f1Z3Kv
Gh0hNtVmXrkMO5TjYW7dOb4FhJuUGqK4s/LTr1wDnM3n9qFgmWz5s56OJaqs9HPzKoUpjR6bQiN2JTrZP2HCppz8Xe9u
g2etvxyA2M8PBpJX5q6GQwMQ+qAhnOkH99UZ7mP4tUOyDO9MfMZoYUFHwR1Ad8ctW0B1HJwiUtN9rvtTp7x7XQfvq3YTU
0NmYU7xmSRJ4xfyntOCj9YQvMmPzaQzgW1RyFIuXlzbKyPfwl+CaomOh9fFdvx1RW5+M45sjnoFNn6Qv0iswPaa9QEvn
A2fAx2OyBLn/95xXdmvX2f8HynJYrJndFj6k0bL5BP7oLcoXhGtUC5J7C8IhQJ8I5gqJcAp7fzVKXk3mHVpDnPl1P6hFO3
ttrXHnMp61Ozc4HmXqBZvX1VTcv7IEfQYE+SUdWyOzmvdmE6flWM2jZeRoVyuIvddJKn8bWNVfW7V7DTdLNZFc4y1Lsa
E3Llz15vr66+/jmbNmkW/fv3ilVdeiVtvvTVmzpwVbcnkWe65Z3z//feFxp911llRunTpGDJkSPTt2zfGjBkTf/7znzf

oMZgxY0YMHz48jj322Khdu3Z07tw5vvjii3jttdcyY1atWhWjR4+OPfbYI7bZZpsi99OgQYNo3bp1jBw5MlavXr1
Bt/1LZ555ZpQoUSKGDBkS/fr1i7fffjvatWsXP/74Y2bM4MGDo2PHjlGlSpV49NFH46mnnooaNWrEoYceWqjgjIg
44YQTokmTJvH000/H/fffX+TtNmrUKI455pi4//77Y9WqVQWu69+/f9SvXz+OP/74iIj46KOPYs8994wPPPvgg/v7
3v8dLL70URx55ZPTo0SOzRENExLJly2LOnDlx2WWXxb/+9a94/PHHY//9948TTjghBg0aVCjDDv/71r7jvvvvvi2mu
vjVdeeSUOOOCAX/MQAgC/Q/PmzYuIiBo1akRExNSpUyM/Pz86Z9N1o27ZtvvPnmm+vcT5UqVTJfZL/1lt
RtWrV2HvvvvTNj9tlnn6hateo697OhrLkJAPwm33//fSxevDgaN278q36/bNNmy0bp164iI2GabbWKfffYpcH2PHj2
ia9euMXny5Nhtt90iImLChAkxYcKE9c5K69q1a/To0SM+++yzz+bjwwgszb7jeeOONOouss+L
ss8/O/O6xx67UffjjDPOiPbt28eHH34YO+20UzzyyCPxxz/+scj1Np966qkYNmxYPPvss3HCCSdktu+6666x555
7xsCBA+P888+PTz75JB599NG4+OKLo2/fvhER0aFDh6hbt26ceuqp68WdkbPd22g6HfWqVXHUUUdF3bp1Y8i
QIdGjR48C4w877LC46676MjPmTMnr14isjPz4969eqt87YGDBgQq1evzhyCdMYZZ8RnN90UDz/8cLRt2zYiNvy
50rhx43j77bdjzpw5UatWrfXez1/aY489Csz3P//mnnnWK//faLe+65J6655ppYvHhxXHTUUdl1oGNiDjiiCN
i9913j7/8Sx/x3//+t8A+O3fuXXB0XJsePXpE+Ck/b4h48//aH75XnRHcd+C0dEKqjKpMJ4oSubiOC+hIM4eqkkuU8dZzp4
ySwl06NAhli1bFrfrfcckv06NEjqlecHlWrVYiM0Q0K9atSoOPvjgmDt3bvTr1y/dH9+H2HwvJ6rJjtMoThjmMN6l1u
A4HctSZK45JJ4LLYv/994+WLvVtGRER+fn5ERNStW7fA2Lp16Lp16xY6EmiNH374IW688mXYp/OjTp06hcbWqV/
ncxu/hZmbAEBOO+WUU6JOnToFZm/efffdUbt27aeqbeaeeGmXLLlo2BAwdmtj3++OOxbNmyAmuA7rXXjFw4MD
461//GuPHj48MD88VK12ZdG2bdvG9ttvH4888khMmTIB5xEyF/xHSJDV9l966ouIJMF1V966aWoVq1/LMMbwySNN8KyRy
9ZDZsq2opRfSqqDKnBBR+eWG6ugyqi9mgNmZIxgaHh2XyTAiHWRSi9kuFNnsWSNWeDMRtdRfBCI7lPDMaKpsWWBKXOa3RP
0Y5GSqVwNn+wokCnn3RcOWOe9wVlqDpCxQWlxwdpEWyDE0PwWdIsNKWEJNRddUbtdtEBq36HwfUFMvzbSvFLvL2mhTU0nKwV
kSTKHonfo0CEifioo27VrV+RapOuz5lDVy2vLy7bbbh7PN998M+bMPmROdO3cu8PivXr06DjXro06D
ssJgwYUKBtaYiIv7whz9s0G23aYiy0yeRBcjDkQL52s+n4WfMI/rqTnS+UrPQO/2MovBuCa1HVxJi9JVJs1Z/L3SnMIS
PH5/5/aeffjr222+/qFSpVpKly4dDz/8cJH/Dw866CDFJgCw0S688XmoXsD6Kb58OMqzZ8wJFHqX08MJ4//33iwO6pfvyZ0rXNJS
II4+MFi1aRO/evde5j3X6tVYe11g0tJTWO9jnp4wYXpn4a6byU6zEpPcHDzs9ZVI8/fIIZIxLkuOqoAQ2cC
i1/f8uRo1asQxxxWTgwYNyhwmPHDgwNhrlu/0lnHHe6j2wjSyZv+fOmXWAatSoEaeffvpGfGGAkZYYckgGbc
9998fO+yyw1oPCf7uu+ixxX/ixx9/jDJDlyt5dTpl3xGsYXBbyV+/22x/iNVSPvUd9d6Ao9g+/j8xKNsZ58yx3P4kV15
5Jd5+++2YMGFC1JK5du8gTzfxyn8se3HGrjPqzhpw+ugjrh8WUQuXxxUt68/Ml+DP2XWi8R6rNqeKRVdaUy4/jxJDN3L2SS9XJz5JH94YfaoEJ86uS8BDVQ
XX38d5cuX36D7WJSiZpnWq1rv9E7ta2Ywe39f/Ktl9Glo1ENmlNZTqUNaPoMDfx4JZOtIVfHF2PUmk9jyyDH94YbYO+44Bw1tsH0S8/nAjMaKR2bXTXJd6nEdiD0aFaflvUNXWZl9r33KrN6UBQtoUTd69bbcHU91+VBpFFFBJp1yDXFG1nbD4bzTEYxcNlj7MKK
AIiK6d+8eL7zwQowePbrA0kFr3rv88m1fcxcjo5QgxumMWhUV08s6Ll6qV/6tpuFf0aoslqVtaLVSrqGDYZEPqkrsGJMEJ9fXFgJq+QdJtevHHrPRm84d5bjNZsgGDIR
wdADgNylZsmQcPfcqfPDb8Z///rcCmzzjJix1rUUf4vzzzz8brnlnjkUd/i6dKlsXLlyjjjvvPM26hePZ69E7vR0dgeYbqQFzbxhBzZn3WzUMtH39fbeQCnRmtIRRoEddbfTgmxIeIUVS3cgFTMzJnlbC7pb5dAM45gb3JYoMxCaoxkmTbX6p0UpuuvTcvl//uOmmm9Y0JZl8J3AC1vQjTPVAbQQpJcy3d4+3hZI+v4
Je9a27zWHsq8p9/Lz82PrrbfOXL9y5cr6SkOo8lH5p6u96J3795x1VVXZbavWbdJOAOGM5zHP+29WI8nvZZS5m2JH5Ksm2t222iBjnwVxrcpCjPwUbQJJCqHejvPuvrzJ9uQBXhmK4bJRsZzdTH+U2Yx7bn2P9ftHw2lyXqsZxmdXAlmeZ+fD7
33ChZsmQcdNBB63yuzJgxIyZNmhSHHXbYr85TVCmcn58nKwVDTTb56gfjjcnmNnJ7Wk97WP1TWj++IMxCvEd2KNpkMmyrJjpR+dp7A5WMqA2EwMfsVL9pDc5K1HjbtdAZJH+19f6eFzC9mqJpb9QFYhyaSfmYxg9n9iz+9YuPcqcd7V5zyeX2ooTzt5f7MMRRmO7FE8vxT4FLHT1tmJi0OjtYH5v/M2+9zjSeJNaoZ9b6JTWUL0G7Pho+KjaebgDqs7W4ZFvTb3nOUw0adu+rZMTDo8krdbSHHSwcVuoK9zDUlE/Ewjpz7aoRq9wNL7vn1Ngo8E9qRjWtSxtR2x8JeH6tMuvNRcaPxbLLuHWtLwvO6p5xwFu1MK7pzqadAjjBsU1rhpSVSu9w5Qx5dBUfDt9eaY0ZvMJ1+fwnxMXrL6XCMh5K4p3r2BoVjB+1rf3D6ZqM5Yd97xwBfv+UtQpjjvDr8Q+aPJSSj0pCHMqMRUV0jLKwsfOUEgYgfPeJDvbK8TS/lg65zUNDF2aCxRt9gt5dlfPbm/wv9+Dul5htKuwyqsP

22y2pU6dOsmzZsrWO2WeffZJatWoVGPPmm28mf/jDH5K6desmJUqUSCIiKVeuXPLyyy8X+v0NfZzXjBs+fHhy2mm
nJdWqVUvKly+fHHHEEQUe0zXGjh2bHHnkkUmNGjWS0qVLJ1tvvXVy5JFHJk8//XRmTO/evZOISGbPnr3O2y5Ku3b
tkho1aiSLFy8u8vqpU6cmZ5xxRrL11lsnpUuXTmrXrp3su+++yV//+tcC42655ZakUaNGSdmyZZPmzZsnDz74YCb
Xz0VE0q1bt43OCQD8NkuWLEk++uijZMmSJdjdmOskXZ0Mc1L0n+/9NRAgDksPfffz9zFuo1M/DYcgwaNCg6d+4cV1x
xRYE1nDbGwIEDo2vXrjFhwoTYY489su+220b179+jbt29WswAAm9bSpUtj6tSp0bhx40LrTfLLrbej
j6rB0ACCnffnll/HNN9/EX/7yl9hqq60yh8iwZTn99NNj5syZcdVVV0XFihXXj2muvXakX2XGjBnx1VdfxW233RY
lSpT4TWuHAgCwfiWyHQAAYF1uvPHG6NChCqxuDCefvrpqFChQryYjsYlceeWVkSRJaovviNiJ9OoNSuXbv48MMP47H
HHitwlnsAAIqfw9IBAAAA4FdyWPqmsaGPq5mbAAAAAEAqKTcBAAAA4DdycHTx2tDHJAAAAAL9S6dKlIyJi8eL
FWU6yZVnzeK55fNfHtt99G5cqVIy8vL9txAAAAAMiSJHbUb9/+/ShRYkV9/tkN/EAAAAAMiSJEliwYIFUb9+/ShRYv3zAkuWLBnVqlWLBnVqlWLWbN
mRUREhQoV9Eu/QZIksXjx4pg1ZZSkXjx4pg1a1ZUq1YtSpYtSpYsuc7xTigUETNmzIgGDRpkOwYAAAAAOWL69D7mxzTbbbNDYJEkiPz8
/fvzxx00b6nekWrVqUa9evWm9qXmkDxDlrewYWxfbl5ZNhcjbGLvAmNdXWxwAAAAAHsanlJkiTZDpFt8nTAAAAAOwhpRsfMc8z
8n2+FiAvmVBFFJEOpPDA7XVYD5mnqZdxJamzMHgtsABjmaaHqgCZtC+4yjeinj56oiTnXMezdxzo2ua5ia4U1xzfQFAntjcYwCDs+k8Yy1J
R9g/R5hSrIjXdH51AcKh1gA9q/VcCR7zN1CkuCAAqcgcKh1gA5YhJYLP3uZhoJS79WLUuzHhj42KPHoF/sWZkZJ3QyA6xa0Ssfy8odw0ezpi
ucPEvn1HU7FdmbR5hBUXeUXd3ZzlDRM9iAvmRaEDkK13v9+vbpR0lyoeQEEvnp1FUVsIH0UiJuxrcqO8l2Nx/CJ+/PQ
ZutMK+GabFXV3C6x0xXeHuCdcdRBu6Bjc8QSSjXnz6PUUkRZhaB2A9kcBlfn6h4CAFxt6WP5XM+Zd8e3bTnZLfKbrSZ9NrWU7SZRFb18MvlZ
UDbSnqMUckkZLr1ZOZm1mOq3Ek0gfjTjhkqvmRQ+UIdOfEEjRkKQEAogpsyLEHnsDO93pYOgNLmpMEvawBMwV3d5x/X9u51xfmmZrvn3M0
4RXLbq9GSoVW+mm7dMJ4F9Nv7vZ5a1/N5v2/8wKHCWUA81FsVtSKTNe8e2t9zcO+dtc1AFW2QFJXYlXWHaUH6YRrB1hV/TLVpoXTlm0+ZZ8Zjq
4kp24w0BP6Vfs0zvHyzOXyd1PpIyVv7+lvnTCDaXVvyB5oSRBsmFQcrZSxu1eWsmkl3WwZ8LjuxsiOgYObDv8JTvUbKSb3Cq3+O7pa0sKmaXbU
5W2N0K/W3wJiCMcR3ZzfXsiSjuzj6o0r5XSDhOMYMkJdsLjaNM0PDgNbf+o/4l5fMQ7rjT6XFxCRYF7VMt24L1pwUUctcgHJULQFd7P3WAXx1KD
ZtarSpIvoK+BzMDB2U3vQDS+dQFbuVbcdBubV5cBCDQnzTjFEk4ATVB0RJbTpEvUfGL4/dQF2v6QhJhXjoTg5WfPBv3hzn5ZzmGrCftGR0s9u
J+KUmO3nMfOvC/kuMMRwJVX2QxlEwdqB2tgQ19dE7sYzfMEl3Lx7l1aNlKKEXJ6ZnNmUgyZnR91yoFxFNTJA9Vnz7xSXvUlfX25vMqBt0YAKpDg
Sk04XKeRjLtUtGyF8WuRrV7ExsfvFwvtuoQ8G5ELpjwH4ajnRzjWW2UGOKEGuE+G6uUyaV2SVrOvvb7z5tQKkiUS7Bo2QTmk2I3ChvOzkOt
8e6VGL8Y7WRI8Nkx4XJ/mbvyQjiH0wozakWNl9lTunOj6/OFS0ZCxQc8hTUSPz4SaVFZf1SWn+zpOyFehJS7kj1xxM/y9zz1LwK+7IXsXAXy
r2znE6vmQz6Mx5pd1OUuqWQOAjmeRGnO6R0yjTzFrBvGJ6upv9hvnsR1qO1O1dOzTqkO6kOqBUnPCzbqUyWNh5Tq1dQPJ4DCnE5bm4Ml3+ojlPm2
LdAeAVL7c2N/5zZ6DVJYr9sWQuq/0QlH+G1EAjTAOyxNcVRbVzLaZb7ykcM5Emfp63p8wFOymF1pjTOjHE3yEEf6wA3aPg26W9ZldYN6e0MfLV1
7pFrFQxhDl8I4iYWqFSsA3Xs0T6ZqKPP7zqAJbMNTm4YUYCSsYmtCJxJrNpEmi9s3aI1ILExlrOsQK7JFIF6QUU7tgFpJyQPzyTbwQLwxMLXi
x7D6w1T+mGiXkYD+BgcwJnhn/9+ZGlS3Yn4fW5obsYpIGVBjfQsgGyW8t3B8eADDaWRxoLT4UtTIqkmBhMJHJiyeX0XE+HfbBd3cRzh4UG/vsPBl
8G0+ZZxJ9MhlJqOVRJFnjmsJz46wf5bpLG1VoX0gv7sbDmD+wsq1wTYMxXtTgfvrC7CGcgUzGnaRNuKhQdYfCFa9H+/5VdDE5M8YBbW5zX1+VhKL
QJf+/7b3wSrvEvtLEnEvA9BPjyn2xQkwDFqCr8Vj7jP7w6hiLpVeVcNsZMk2kA+iGW3Y9EdsNIFLPD3H4/a4gIQ0cFKVRj6NBWJa/dzdp1SA/0
/FZzSCOHGQ63+zLl+AQCZXxKoE/F8ML8zDNEK4XQVXe/ozPGYGBxzOM0qWxTLbtnJr/kRgs+rZ6ncmAJFD4dcyeX8eV+XXwGIV/JqmdI7Q2BA
V/O32NaxYt0vRpr+GpT/R2jOg33+0pEFFe6vcB8+1yGdg5RBTgI4tmRQP98j+iVgEg5aGdNG6zjCBqwnFJ2DGVoqc1qgQTYd/jYwI+aJAY
CTpqyzw0mSrXsZZ7b8gzqjVbcebMGCyuNGNrz4AzZcSXunuUyIs4a9qQrWjH+8Hq4z4fX8k7SyvBMR26BB2+QaRiMbRbW8fWH9tmsPAl3/F9ff
T6/eG2yGnTXGi5f0czy96DY8GP/zL8YWo03AoHM+XgSoxYHdDYUqtAvPzFuzhzIq1nzeUR9gTZlyyJqjWBHlhRPfNW7kqD8KzSpH33qm1rTtjf
6u13rRrQTZQrMaeVBgfBV6qUWTD7P8v9YK6ce4f6N+/nY2rDrK5fFFrB+GKZqKCp4ezPaGqd+SLdmrGUaDpnLk1HyVjM7mTPMg2RMAZHm0C2u
ppz1LT1fnvz1PLBBvqO3RWSCaFLr1rq3PuyOqiPoFzGuGaSJYp7eF4L7HaR3f7C0IOuMnX29uVvTOPz7cpFfJ8fX6Yq3SL/oWlCN0r7FL0/n2jL
1rM5tTEQk7OuXhJt8O8FLC0VtqHm+uN7BR0dQYlk6P8P7KtcB2zrz/fz5GczO/8zvtjwg7Qk3lHuX8u0W25n3wtD9nHxz7F/V2N9Q2KZd2uO
8LL0tjy2T0rP8lQOGSp3V3Cm77PPnn5n7vH+zevWpc2lWY/2UcZz/5j7p6VNDUIvfFBlMDP2wnvzL7v3x3bNKlL3Wp8GgSv3fhW3xj8T/9GK/
nz3xrlvL9P5QUnP3X3LGsl1pjDz4aeb6nJGsJ0t5P/7H4VQwJcz8n9fJ1zuK6NO4A3JC5gN6jvS4rMnE9ta5kLQuGl1sDGV1z6c8P6zrNv3wLGx
8uN8E3d6MJrfPHq67nKy1L3KJrxG29+fgGk7PJf9/1K8ydaN6ax3T8nn8CuvbJPw8k9v3HWNYvbt7c/7b3fdNjJa/YwvcTBq6PzV2P5i6QxZ+
zj1n2v1QX8vh5/9+PFNNY6f7e5PdWHr3v1OnUE6/WP9Rn/8w9/O3p7/7Tu0uP8a/cnbmP7bXnOenzSpn3R2/PX4jHD4Ai8wOnl9Jzr6aNt0r5N+
0u9P9fP7mD6eFi5+vLfNnvz6LnYzy89PU96d1a3ffX8WvDtzt/0fN/4nH6N97G9Ku2Vv2n5nFsTuV3lHO8gH9b6xKHbHb39F5ov+PrzNbLL
bzJgyZcoUmPG5Zsya3y/KzTffHNdff/1vjQ8AAAAAZFHWZm5Onrr4S49jKo8fF/9+8f9Nt9RXGLs5iwAS6L7Fjd9A9+7H3rvtr3e3s0+Tz3+P
7wF+uZ0nZC2N4l1Tqy/MyryNudxDR1H+d0+x9GSbOwSgGBtqfA1HgcZNW/7Bkpef6E9Jq+EvzxOi6EW9gSpvQVO1DjfmNU9GP7wCHKzrepb
22KLBpT4cFfRlU8prDevfXYqqMTmTgt5SV3FGV5UQmjhx2+bV/Wvau3yX8GOd3bQezVQpHw3v8hZmqC8WjtFTB7YP5r84wvlzH8Ys62qXSUx
I7MJz7fFcj4IxX2gu5uy9F4uOFpe9Sa+/lpHUpOvZD1ljJ0wz1m8sLuH7i7PPaYQ5eTwbOd2v8sE5s+vOtrMnj3YkuiHJOXmpvrvdBYS7fl5zP7
oPr7eF1sHnTvz8m2x2F4qeUN2tFiQaY9lfl3t4t5pzLq2yyu9LBe+pxY5yvh1L8t9xfv6RtY70r2vlxT30vf+a9w3vzQdSEN06m4KtYCqr5Uc
YnrNw27eC7jDwlfVKRWtbvHpt7qP7kn2GQ9mFH5wWm7ZvvPHmjp8k0/HJnHRgbmmz+/x1uP+3Dc5+fz4+ZJPK5EG7rb+0zcL9sjv8D6e4L4u
T9oLLOyjl13PuvO7xw+SXHPkdUTWo0Vy2EfzhVZ2ZzVePCVwnx2PEOWZY/x/qxX7NH4f2rU1kK3qvvXzdDymXKr1rumtbNfUw9+C1GZKXZD3u
69GtqkwHfcSp6vpLpXptSQo0f0bbsw7WNj83+9W71fR9b1cYV1f1LtfP1q9uzvd1bK/3H3zv9jXkGVd6WwtDczg9xq47vLwuhP+ryYfbrtDj
fbQvdtT3Y6tKsUkGxv3o8WzrwaLZsn1atrb7xkFPfz8rZZvH8fHQ2jcb8dv5EJ5sr9eD3Wc98fHQXCFvnt7qNVWbL8NTbsVFm/XcHHtf7c7wb
3qn+PAz9e9exWTy3nZqKHDdjyc+zyMhl12F2ZsPmLnXlN+UVUN6HddE26jdTGwcsoUZtZrff+PTrzXtfTd4XtdvnNXaHpS7ZU/q9RTkbx/7W5
jt2t9bdW19f/LtL5xOOZrY7eQvsqKdIjMvzi/DCTzjZfxvNRMzp3ODxzk5Tfx9z6jLT9zTxp1dr7rDbcrcMB9efX3NvNWGu3gLI4S3d3Ee9o
38e+vGtR8+vnW5V4Z6N6tz+rqX+wL9L+d5l9kb3vDOtsO3C7Mn53wnx1uazKn0Xq/9Lmm8HjBp9fhxr5t5rbPqW+2zM58fnPDf92mb9e21b
+5WT58ROlcLmrYy98yDq9e2KHBtvgRVMnjTt8f15m7m2jZdYnk57zMNlXfwH0k8D8Q3Ln+33v3Npf4wOz5Ufz38DXeq7m3aab7n3j7Y4kNJXX
gaVJfXL9ZPd5t1W57fT5b7zuTHv/2T5+v7zXPj1D3u92XMl+5O6PTpN+Au7ttD3z5a1vm7hfv1l5MLWfO5jxb4cB2PSWh5r9h+jP7v+OZJvjsD0
vvTK3Cqz9W7ZGMMsOeo8WCZIRJgT1vbuaW5t8Z9b4GPyWdwmXjw2Oy9O7y+a6eV5XD0jfOM12tAjnLnr+NddHPHmhj4xqdFuOVvjGx9zX5LT
v+v5b39nG/+HNLfnetmaozH6VRQD0Zbv8+s9aJ5Hr3jsZrl4+W3Jl1j3tP5nnYbO+q3PNzXT87jv5+vD2LiOZlzeY0bZ3VnKOt29oB7Tv1/z1Se
vOXjuz2HnUx07/z1nfwPDHr9XZhrMr2Lql3aj2Jiyu3n3xWfL3P9yKmC+PwP/9/y+P7D/m5PfVL28Z8k+vXXb3J6a8Hnq1Hn7vO5Y0fXpp+bP
8fq4bdNHvz+e9/sPM0W5TwqCBpjv8f4bN/Fs3sPPEJ9YVa7CX2z2CnrCsFpnkkSqvNyvL/jp/vXbXFUo
1r7G0LHdNr/CD/Pcm9Hl4cv+PrPbu11X+fBRY3O5/nPF/PpcLPPRdK3lpZ9FvtzTt7VFmmY3+K6XMzs5sP
9OJ5Nc1r/b45v1PvcsPrHHA8s5wPHLxZg6vtqBxTVrrrNdXA7vDfd63fVtf3rFH/vbfCKm1L2vq/oPT4rc
1pn27HLt7cHzHNbrbnXw4XHbMvPw8yHnV8yh8e1HV8m/vWIJt7MY8c7O9jMbxrn6HxvuOQ3e/v7bqVXPgr
vRT/2dVj/yrmaDDq/e15POzLNP9v0cXv9n92zf/1L/9mdbFb17B7t/L/xTLoXvdy3/3YgeJXNb3f9uPrtl
8+Jz9rafvb0mdO0Z/RvUcOMTDy/Uh0fttcXxL3VjSnzk/j9gn9dkWzfHpl6c9ZqV8ljgvPvtmzfTz7mWjm
TnVHffbMvqxvP5L5SRv9Xdcf9+dH3Sqk4x5fmpbR3ydev5YdUBpl1l+3Gf7lNb3Hfjl+ebXzH+0yZXfNxT
+VVl5WvHP/3dTbTu+xzmnjcvGx8Mn+p+HLb4fMbGT0Hq3SWq1+m18sf9pTqwMdcxNzNUcyrfXVLkS/f73Z
kPeXn91+xfGq+1WfT3L/8kNZf5PpY7lvnnOqN0m3k+Tqd1bzDpYj+Xk0GzK9+OdrhazW9HXkm7hPfJu9Lw
zyJ12tHsHf5zb3V1/N7hN4+GVrqC1Ga/rZwGfr3NmvbY8RwZuz95zvF3RGfcOwXvi+zuqXrYTdzO82X5bb
vOn3/b3t99f2S7N9mP1DsDHbzTHWgcNa4lbvvOuwMtrd1TpmTW3fe2Pn9dWmN+kLiWd91RPvmufQ+7vkL
HuN3hOWzpz9t9cXDuPnGe0MubbWNPJ3Vh8ivrOTO9Wtn16Ir3WnOy8Lme9o5PvuGe+/fv6hRbp6r+tYVX
1xvPbHNuFfbt/P/N9rPGOnIf+UO9TruLDP3sN+ynvR56Ntfdd/3O/ux/qtne1W9+5n/P/ch+3fH5xvbL
1NU3tdzrfne6T5FPttdqzmHYPyblNk+0m/zOtPb+9pnd+5mCWuhN9WNfzjfaE9etXrTl/8WFY1TNZnDbjf
29/jPPWXnyXv7vD/+Wv2M97fHLrbX1jYNGbmHn2l1rqfK6xx7Wrzbnc4TNHjlcfJ+1DbNnFfhfl7ny3bn
FtZ9T71+vcr3j+jtNv3X6ZJ+O2/33nWpZ7UPu/n0PpcbHWzBHqvW5eEgnlGh4Lcg8vpvYdRfv0Hx/Ln
6r5UH2fk9+PH+PdPbxTmH7mEv7+1PP1Mu1+z6p/7uI2dcwYxD7O5w57d/Jh8t5bP9rXu0C/hR/Bh/3R2
+DoeP9v3zZf7QPP2+9xXbZHgbJ7ffcW2r7v0cHoIM6c/E5jZnTmb7xVLsPqcvNLxn3u6P3Xd9LUv+T7
G2+/yrqmWhN+D4fZnnqL5vdy0JY3fUwUa39TEQ39C7q4sF+GvzfVKsULAI8i+FuGEXE6+FrXuYUZPCJC
ae1Lb7Z9Xn8qTsvMw9GXNjfH8Nf5rbO6bT3MMdfZ8Pnb9v7+VH4Pc3U4TaMvPmtXXl47vP0tFd0g8b3Q
w8vV2W+bn92c8X7Px+XzqgYv9H2zVMZ5l6J89Nft3f3dWr3p2/Wv+w3+5Tvf7O9vvvT6/Ls1XVfsf3U
2e8ftTnX9bD9X9xNdwcLxZ3e+vz5fw2fz3+FKe9bpKX9+n2PT/nDf4o8f+n69/bKvntfF9vv0KfrPfbxNO/W
wX5Pz/cvlxvb8O3H6MfvzL3jqcH0r7NOvp7NtbF7ddv/Z7Pfm/vPdNRqXKgH9wLDL9OtKfkvnz2pOzU
9ZflEb3PnzVx9P+B3i7SVW6Kfa+GeOKf0e72f5fntmfn+Lm/WtxHXafX3dB5U/vXtL9Mz7w+xnz/3fnfnZ
xhg9WcXYYX2+LkLz2V7X9vbuvPnJ9p//fu3a7++Pzpx3f/dJVPpvs9dm7V/hN7V1nn9ev/Ppr+O3XPwJ
2u6PqfFS+6OZ29Mb/3s0pO/L32N8O4hfz7+P/e/VZqr9VcfrvVVd3f2fy6rtq3Fnd5ldzvY5vX4+8PzLX
nWfrxgfHL/tfZ23bXr7z/mvrW5UaxHYHfS/e6/mL0Tt6UA/p4un4xZ4AX9O8a+Pa2vO2y+3W+Jv3t91/
Kff51b7h0P5pfH7z3dd7fcU7nLP+b69/B1X3j1wa7O/nWtj7fL3vVt/0bbbX+v0f3B+8Xr5rtVm0sHxX5
7M3LuY7fXt/mFnx5oT1E3o/6v+f3pqP8XsdQzXPfuPcqW6u3+/vsbPXXwqzZ/v8Jn1Q3y/y6vPH/O8Cwb
Wqa1TN3zGxN27H/Pbxt/f7z0r3z5u2H/P7+X51fzS9/3nj9XfGVuNmv3fLf+c5m+fnoe4Pb+fP5XA+q5
j8XWkP8tv7HFzzX5z7+/f7/mTzKf95vH6Rv3V4n+rNXn+5X9zf3fq8vt+fc4fXb7Xv7/v7a3rbj9kPsf
vnf3T8/Pwb+f3nvPEuwf9Mfn9TLc+Xx1vz+Xf8OeP87fb9Wf+7/vx3U3W3Pvv3+fz+XX3v8qXTP+P4ezV
uLPbu7y3q/PhtfH83fn+X5L5f+HrGW17T5r+vu/Vp3v71n3frr6vffPe+7W/8fyy+3b/en3v13X3F++XbP
A9f9Lf+H9P2uv2y3VT/RGXn9tvXy3xVvnfr8++5Pf57N+d8p+nv/s7+n/vLf87/v9x3N/+Z3f5/n+r+0
a1/2X5LvPvt7fn6m/93vnX5mfn+3v1Z/1nL9z3n7Pn+rTvLXfb9m/x1P7e33+dP/+3f3/5f7+dX8//2Pvf7/
8c7/Pr/+f/vv/d6/v8vvn8ff/b8eX+/X+//fP/x/7//+z3//b7+f//P7///c/e//n//9v//+/6/+//v/b
+9/n/r//3+//D/vv+9/u/f/+/7/bv++/r///2/v/P/7/f+/9/7/X//+//+/3//3/v/v/+/9/+/////v/
22y2pU6dOsmzZsrWO2WeffZJatWoVGPPmm28mf/jDH5K6desmJUqUSCIiKVeuXPLyyy8X+v0NfZzXjBs+fHhy2mm
nJdWqVUvKly+fHHHEEQUe0zXGjh2bHHnkkUmNGjWS0qVLJ1tvvXVy5JFHJk8//XRmTO/evZOISGbPnr3O2y5Ku3b
tkho1aiSLFy8u8vqpU6cmZ5xxRrL11lsnpUuXTmrXrp3su+++yV//+tcC42655ZakUaNGSdmyZZPmzZsnDz74YCb
Xz0VE0q1bt43OCQD8NkuWLEk++uijZMmSJdjdmOskXZ0Mc1L0n+/9NRAgDksPfffz9zFuo1M/DYcgwaNCg6d+4cV1x
xRYE1nDbGwIEDo2vXrjFhwoTYY489su+220b179+jbt29WswAAm9bSpUtj6tSp0bhx40LrTfLLrbej
j6rB0ACCnffnll/HNN9/EX/7yl9hqq60yh8iwZTn99NNj5syZcdVVV0XFihXXj2muvXakX2XGjBnx1VdfxW233RY
lSpT4TWuHAgCwfiWyHQAAYF1uvPHG6NChCqxuDCefvrpqFChQryYjsYlceeWVkSRJaovviNiJ9OoNSuXbv48MMP47H
HHitwlnsAAIqfw9IBAAAA4FdyWPqmsaGPq5mbAAAAAEAqKTcBAAAA4DdycHTx2tDHJAAAAAL9S6dKlIyJi8eL

v3ryoXbt2/POf/4yTTz45IiK+/fbbaNCgQfz73/+OQw89dINuc/78+VG1atWYN29eVKlSJSIiWl8+qFjuz6TbTi+
W/QAAAACw6RXVE5G7cmbNzVWrVsUTTzwRixYtijZt2sTUqVMjPz8/OnbsmBlTtmzZaNu2bbz55psRETFp0qRYsWJ
FgTH169ePli1bZsYUZdmyZTF//vwCFwAAAAAgXbJebk6ZMiUqVaoUZcuWjfPOOy+GDh0aLVq0iPz8/IiIqFu3boH
xdevWzVyXn58fZcqUierVq691TFFuvvnmqFq1aubSoEGDYr5XAAAAAMCmlvVys1mzZjF58uH/++dG5c+f
46KOPMtfn5eUVGJ8kSaFtv7S+MVdffXXMmzcvc5k+ffpvuxMAAAAAwGaX9XKzTJky0aRJk9hjjz3i5ptvjl133TX
uvPPOqFevXkREoRmYs2bNyszmrFevXs271jFFKVu2bOYM7WsuAAAAAEC6ZL3c/KUkSWLZsmXRuHHjqFe
vXowYMSJz3fLLy2Ps2LGx7777RkRE69ato3Tp0gXGzJw5Mz744IPMGAAAAABgy1Qqmzf+l7/8JQ4//PBo0KBBLFi
wIJ544okYM2ZMDBs2LPLy8qJnz57Rp0+faNq0aTRt2jT69OkTFSpUiE6dOkVERNWqVePMM8+MSy+9NGrWrBk1atS
Iyy67LHbeeec45JBDsnnXAAAAAIBNLKvl5nfffRennXZjw5M6pWrRq77LJLDBs2LDp06BAREVdccUUsWbIkLrj
ggpg7d27svffeMXz48KhcuXJmH3fccUeUKlUqTjrppFiyZEkcfPDBMXDgwChZsmS27hYAAAAAsBnkJUmSZDtEts2
fPz+qVq0a8+bNy6y/2frYQcWy70m3nV4s+wEAAABg0yuqJyJ35dyamwAAAAAAG0K5CQAAAACkknITAAAAAEgl5SY
AAAAAkErKTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgA
AAACQSspNAAAAACCVlJsAAAAAQCopNwEAAACAVFJuAgAAAACppNwEAAAAAFJJuQkAAAAApJJyEwAAAABIJeUmAAA
AAJBKyk0AAAAAIJWUmwAAAABAKik3AQAAAIBUUm4CAAAAAKmk3AQAAAAAUkm5CQAAAACkknITAAAAAEgl5SYAAAA
AkErKTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgAAAAC
QSspNAAAAACCVlJsAAAAAQCopNwEAAACAVFJuAgAAAACppNwEAAAAAFJJuQkAAAAApJJyEwAAAABIJeUmAAAAAJB
Kyk0AAAAAIJWUmwAAAABAKik3AQAAAIBUUm4CAAAAAKmk3AQAAAAAUkm5CQAAAACkknITAAAAAEgl5SYAAAAAkEr
KTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgAAAACQSsp
NAAAAACCVlJsAAAAAQCopNwEAAACAVCq1MYNbtWoVeXl56x33zjvv/OplAAAAAAAAbYqPKzeOOO24TxQAAAAAAA2Dg
bVW727t17U+UAAAAAAANgoG1Vu/tz7778fn332WZQpUyZ22GGH2HHHHYszFwAAAADAOm10ufn222/HmWeeGR999FE
kSRIREXl5ebHnnnvGwIEDMyXnnDlzolai6KOP4uCDD47y5cvH4MGD45133qolJkybFP//
5z1i1alXsu+++8e2338a9994b995776bKDAAAAACw8WtudujQIZ599tkCZ01v1apVnHLKKXHCCSdE+/btY/r06fG
f//yn2MMCAAAAAKyxUeXmmDFj4j4//8U+Mtf/hJ77713/Oc//4m2bdsWW0gAAAAAgF/aqMPPSFyxYYEHX
r1l3r9fXq1YvSpUvHoYce+puDAQAAACsy0aVm40aNYq33357rdf//97//S3/97//T/S/7dr3+q/7e/e/8//vjn+/99//jD93//
SS+KDDz4odN2UKVSsssuiz/96U/FFg4AAAAAYG02as3N2QGJI/2drdIaG4HEBm7
G1VdfvUmCAgAAAD83EbN3CxXrlyMHj06HrDffdlpvVn5c2bcf//9g3bcf//9cf/998fMmTPjr3/9a4wdwOzY+/fTTTZUVAAAAACB
jo8rNiIgygZcrElVdeGZMnT4xH4sWL/FixfH4sWLY+zYsVgSpVo06ZC0q6GXkmGjYVsaTNWxSSCqNXGMZnTtgYgQUNmrRJ0zsZy5599Nlq0aLEpmMgIAAAA
AFLJRMzePOOKIaNGi0xzdNmTKIaNGiRXz00Udx9913x7x7x7fffht3333333psoGAAAAALBWGzVzc/jw4dGjR484//zzGjR484//zzo2nTppsqEwAAAAD
Aem3UzM3XX389FixYEHvssUfsvddecffdd8e33367KabFlYFixYEHvssUfsvffe0b9//5g9e/amygYAAAAsFYbVW62adMmHzzwQ+fPn66bTT4TT8TW28
dq1evjhEjRsSCBQs2VU4AAAAAgAJ+1dnyK1SoEF27do033ngjpkyZEpdeemnccsstUadOnTjqqKOKOyMAAAAAQCG
/qtz8uWbNmsXf/vrWmTFjxjz++OPFkQkAAAAAYL1+c7m5RsmSJePoo4+OZ599trh2CQAAAACwVsVWbgIAAAAAbE7KTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAA
AAKRSVsvNm2++Ofbcc8+oXLly1KlTJ44++uj4+OOPC4xJkiR69+4d9evXj/LlQ9//qF+/fpQvXz/du0w58+YYEXy5Yti+7du0e
tWrWiYsWKcwxx8SMGTM2510BAAAAADazR9DazrJabY8eOjW7dusX48eNjxIgRsXLlymjfvn0sWrQoM+Zvf/tb9+vWLu+++OyZMmBD16tWLQ9oM6Zv375x++2RBNR+/23R//
+/WPChAlRr1696/NChQyYcA7zpmfPnjF06NB44okn4o033oiFCxdGx44dY9Wr8MPgEbe+ugbNC6DwDwHEeR6qqcvvFYdg9bIz4hIi0oktq9MEhIEeRA+ijc4h4
PGDAg6tSpE5MmTYoaDDzwwwkiSJfv36xTXXXXBMnnHBCREQ8+uijUbdu3RgyZEiceeJUCSM2JgDSAyeaiMZJxT4fYUM4fMDPeWHUi0t3BHj++9z+xccyyCE
RETF48OBo0KBBjBw5Mg499NNfr8AAAAAgE0vp9bcnDdvXkRrE1KhRIYIiip6dGvn5+dGxY3ztoZp0bbJqLJ8dqwYvYpt2nl0smn1qfj8bDzA6uMjdWzTf
fjIiISZMmxYoVKwqMVrRs2Tl2peWLVsW8+fPv+frRsAf3TZ8aXy3+fR5+33Z0fkASPwLPeI+gm3mMAfhBQGmAPb3AEBIZ6l8gEINJ9sPEFkkrZ3AjkC
3bt0CY+/vWrZu5Lj8/P8qUKVFq1d65hfuvnnm6Nq1aqZPZV+J6GDYr77gAAAAAm1jO0lJsXXnhhvvP/++/H4448Xuix
vL6/Az0mSFNr2S+sac/XVV8e8efMyl+nTp//64hHnN/ts1tYhEeHzqiwBMAAAABAVuREudm9e+/d44YUXYvTo0bhNNttktterVy8IZ
zZmVmc9arVy+WL18ec+fOXeuYXpbtmxUqVKlKWlwAUAAAAASJeslptJksFF14Yzz33XLz66qvRuHHjAtc3btw46tWrFyNGjMhsW758eYwdOzb23XffiIh
rFyNGjMhsW758eYwdOzb23XffiIho3bp1lC5dusCYmTNnxg8v7zy7+/R7mw3Dx+e8oZ1jHuSI/Ho4gVcA
nZmhWrVo1ypcvH3l5edGzZ8/o06dPNNG3aNJo2bRpg2Rp9+vSJbgwv8q5xmoBh3g+vSJChUqRKdOnTTJjzzzzsesekqzBbSpjyt/qdlqdqjdqjdDdtf6s
3zpw9HQAAAADY8mS13LzvvvshuIqJdu3ZRrFiirFmy6hpfqFBT7pl3KH5+F/3/qLo3v87/mG3+C1AlqOBSA8LRj6+
dd9wRpUqVipNOOqxmHbeWzm1+2vrcR6B3l8/IVB1cLPM3BTgrW3n+LqAeeo5XhYAhqeWvdlNN5QoOtpNGL2NO00oxs7+cjAAjW13x/XCyfpT+R/bHJ
Jt51eLPsBgE3Nax8AABTdE5G7cuKEQgAAAAAAG0u5CQAAAACkUlbX3AQAAGDLYYkTADY3MzcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgAAAACQSspNAAAAACCVlJsAAAAAQCopNwEAAACAVFJuAgAAAAC
pVCrbAQAAAIAN0/ryQcWyn0m3nV4s+wHINjM3AQAAAIBUUm4CAAAAAKmk3AQAAAAAUkm5CQAAAACkknITAAAAAEg
l5SYAAAAAkErKTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCX
lJgAAAACQSqWyHQAA4Pek9eWDimU/k247vVj2AwAAaWbmJgAAAACQSspNAAAAACCVlJsAAAAAQCopNwEAAACAVFJ

```
uAgAAAACppNwEAAAAAFJJuQkAAAAApJJyEwAAAABIJeUmAAAAAJBKyk0AAAAAIJWUmwAAAABAKik3AQAAAIBUUm4
CAAAAAKmk3AQAAAAAUkm5CQAAAACkknITAAAAAEgl5SYAAAAAkErKTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgI
AAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgAAAACQSspNAAAAAACCVlJsAAAAAQCopNwEAAACAVFJuAgA
AAACppNwEAAAAAFJJuQkAAAAApFKpbAcAALYs027YuVj20/DaKcWyH3JHcTw3PC8AAPg55SYAAAD8zvgyEthSKDc
3MS8YsPFaXz6oWPYz6bbTi2U/AAAAQG6y5iYAAAAAkErKTQAAAAAglZSbAAAAAEAqKTcBAAAAgFRSbgIAAAAAqaT
cBAAAAABSqVS2AwAAsPGm3bBzseyn4bVTimU/AACQDWZuAgAAAACppNwEAAAAAFJJuQkAAAAApJI1NwEAgKyxfiw
A8FuYuQkAAAAApJKZmwAAAOQUM3oB2FBmbgIAAAAAqaTcBAAAAABSSbkJAAAAAKSSchMAAAAASCXlJgAAAACQSsp
NAAAAAACCVlJsAAAAAQCopNwEAAACAVFJuAgAAAACppNwEAAAAAFJJuQkAAAAApJJyEwAAAABIpVLZDgDA/5l2w86
/eR8Nr51SDEkAAAAg95m5CQAAAACkknITAAAAAEgl5SYAAAAAkErW3AQAAAD4mdaXDyqW/Uy67fRi2Q+wdmZuAgA
AAACppNwEAAAAAFJJuQkAAAAApJJyEwAAAABIJeUmAAAAAJBKyk0AAAAAIJWUmwAAAABAKpXKdgAAAACALdG0G3Y
ulv00vHZKsewHtkTKTYBi0PryQcWyn6GVi2U3AAAA8LvgsHQAAAAAIJXM3AQAADaaoxYAgFxg5iYAAAAAkErKTQA
AAAAglZSbAAAAAEAqWXMT2GJNu2HnYtlPw2unFMt+AAAAgOJl5iYAAAAAkEpmbgIAQJYV15nHJ912erHsBwAgLcz
cBAAAAABSSbkJAAAAAKSSchMAAAAASCVrbgL8DlnbDQAAgC2BchMAALYQ027YuVj20/DaKcWyHwBYGxMuKC4OSwc
AAAAAUkm5CQAAAACkknITAAAAAEilrJabr732Whx99NFRv379yMvLi3/9618Frk+SJK677rqoX79+lC9fPtq1axc
ffvhhgTHLli2L7t27R61ataJixYpxzDHHxIwZMzbjvQAAAAAAsiGr5eaiRYti1113jf79+xd5fd++feP222+P/v3
7x4QJE6JevXRoUOHWLBgQWZMz549Y+jQofHEE0/EuHHjYuHChXXHUUUfFqlWrNtfdAAAAAACyIKtnSz/88MPPj8MM
PL/K6JEmiX79+cc0118QJJ5wQERGPPvo1K1bN4YMGRLnnnntuzJs3Lx5++OH45z//GYcccckhERAwePDgaNGgQI0e
OjEMPPXSz3RcAAAAAYPPK2TU3p06dGvn5+dGxvn5+dGxY8fMtrJly0bbtm3jzTffjIiISZMmxYoVKwqMqV+/frRs2TIzpij
Lli2L+fPnF7gAAAAAAOmSs+Vmfn5+RETUrVu3wPa6detmrsvPz2zZcpE9erV1zqmKDfffHNUrVo1c2nQoEExpwc
AAAAANrWcLTfXyMvLK/BzkiSFtv3S+sZcffXVMW/evMxl+vvTpxZIVAAAAANh8srrm5rrUq1cvIn6anbnVltlts+
aNSszm7NevXq99913wd+FzmoteNW0xCMTGTh1WpUB/4uDm4F/d7+wtpP9+PVoT4/L2Tbxmfzj2gV2QW2gi2zpnJ+
aNSszm7NevXqxfPnymDt3boHZm7NmzYp99913rfsuW7Zs1C1bdhMlBwAAtlTTbti5WPbT8NopxbIfAPi9y9mZm40
bN4569erFiBEjMtuWL18eY8eOzRSXrVu3jtKlSxcYM3PmzPjggw/WW/evW12uB4cxcYjq4zNmO2zY9N1e5mtnl3gzJjh4w0Vw7YG4o3ZmnmxnC9b0D
y5MlRo0aNaNiwYfTs2TP69OkjTTZs2jaZNm0afPn2iQoUK0alTp4iIqFq1apx55plx/vDwY1pUJKPyuL9+5k6PC+8Z5WF7NCcwCy8aCHi3YzhfX9i2ix4CqC+lvnHgv/gdmi6Sa3HRbY+fagbG3OHd4Dlc3OOlwkJSK5rGwSRGr5W+VJ0rFWMrNa5CvmF8+U7tl1W9mLzv5/I3m7+3HV+98z5W1+xFq2RWfamS0dFQXz4wi95vSzwoY1eNwBtGFeHE3+8y+U7dDD4PrrWTtQ6+sLX7tn17+b4MAbhWeM9N2vl9RzmZ0WbMD6tmU/FoBE5lBNXfjx3dHsFZqa77q+65wOGvYgaFNTa2x7yPfo9DPnuLx7g0zCNR0EOSgIa8fhNDHPhrk5+fm6AQQKuHvzTYiLDAvK5zjW0cj7zDmRhFLPWb4oGWzJZ2cbWP7QNtYkwS4RdLgAyyAwEGLGRhuOr7vMvd2uUj2UVe9ONaDD5mZzCe3uhj1oGmYzO2NCckLbmeV6ctC6bm9V6AGKPLzj9DNkIHSEYo7SV5kDEXpvzRtFlRvd1B1zdC8+lhtKC2Ut1f3yCtIh5fJIhfuTgL7F5PIMWcqTXoRv30ckufZfVD8AugFPTfDdxO8J9l4N62N9yRnsxSCVYfC2N0CIKL0pm9e0lkMq0jW20OVyD2WaZgxgIEP0HRkDJi7fdZKnn1wPfFJGxKK2cbHtFiZ9x+Fm2dkpDnW8OtxnMnYPzi1b4aLaoXMZFk36yR1Zt2
```

zSFveCJk3h7TljZB5c0lb5rTljUhnZopf6s+WDgAAAAD8PqV+zU0AAAAA4PdJuQkAAAAApJJyEwAAAABIJeUmAAC
wSa1cuTKuv/76mD59erajAGwS/s5B9ig3YRNbsWJFdO3aNb766qtsRwEAtgArVqyI7bbbLj766KNsR9lgpUqVitt
uuy1WrVqV7SgAm4S/c5A9pbIdgM3vhRde2OCxxxzzCZM8vtQunTpGDp0aPTq1SvbUaBYrF69Or744ouYNWtWrF6
9usB1Bx54YJZSbVlWrlwZY8aMiS+//DI6deoUlStXjm+//TaqVKkSlSpVynY8IMtKly4dy5Yti7y8vGxH2SiHHHJ
IjBkzJrp06ZLtKACbRNr+zv3www9x7bXXXxujRo4t8bz9nzpwsJdswCxcuLJS5SpUqWUpDNik3f6NWrVpt8BvLd95
5ZxOn2TDHHXfcBo3Ly8vLmW+datSoEZ999lnUqlUrqlevvs7HPBf/AB9//PHxr3/9Ky655JJsR9lgr7322jqvV2L
9ditXroxy5crF5MmTo2XLltmOs0HGjx8fnTp1im+++SaSJClwXS79zTjhhBNi4MCBUaVKlTjhhBPWOfa5557bTK
2zDfffBOHHXZYTJs2LZYtWxYdOnSIypUrR9++fWPp0qVx//33HmDFjivwAcvvtt2cp1f9J++t1mnT
v3j1uvfXWeOihh6JUqXS8pT/88MPj6quvjg8++CBat24dFSSbKbMvFD//169ePdu3aRbt
27aJt27bRrFmzbEEcq0gsvvBCHH354C5der2TRnL9OZ2G58baXkvy8vKiXLLy0aRJk+jSpUt07do1C+kKS9vfuT/
/+c/x5ZdfxplnnhhlL169ZNxZdmU6dOjQsvvDDGjBkTS5cuzWxPkPo+weaXjnVAO+3lRuHTp0rj33nhjRYsW0aZ
Nm4j4qQj48MMP44vrPnmLAuiXVfxMHf3m6QZbbBedjSW0aZ
9/HHPlReN9X14/rlc+yBdqlSp2HbbbXPmsdwQ5513Xuyxxx7x8ssszvx1ZbbZWz/xarVq2ayVa1atUsp9k4F110Uey
xxx7x3nvvRc2aNTPbjz/++DjrrLOymKywu+66KyzEc1vOS2Fw5HXXXescm2k/59XGpfTmTjz8
SK/8W0/p6vb7n78/lynP5v//9b4waNSqGDx8eO++8c6H3Frn2xU1ExPnnnnx8RRRfxufiBdPfdd49Ro0ZF9erV1zu
hIVcmMayxePHiuOKKK+Kpp56KH374oc56HKmvXo2uFq9x45T9Zz1qbYKqt0VPVh6oRgmGqY0vnfZmrYJZa9TzdT+xrqmZ5hO1KE6y3pSYZRy1r4iIMkqq1JSAFOadpPmWg9r4xIyQ+aDD7Nx3ePjqLeucpD/ZSNfNjSPaojJNUCXuGwrxNo4Xnhfy9pVrjyXT
hIVcmMayxePHiuOKKK+Kpp56KH374odD1ufZYR0T8/e9/j7Fjx8btt98e5513XtStWzfatm2bKTubN2+e7dP1mIlV
nv/z8/KhTp846J43k4nM6In3PjWuvvTZuummmOPzw1xxKaHmFOqfxUiqDI7Nxs3vU0wXZbO5yWF8/jvvMhFfGY
Z2Y6bur9z48aNi3HjxsWuu+6a7a7Sgb7NRTT42IiEceeSQ1hSybXl7yy6k3/GpnnXVWbLXVVnHjjTcW+gPXl2MZz2pvH
6PPLII1lKtnaLFy+OChUq5OyrmBOs+DwXW6M+gqMUnKeW8PDGtgrKsOvGt39
0g8d27tx5Eyb5dQYMGGt/NFUqJHtOOtVsWLFeO+996JkybkZbZjrLFqlWrVrzxxhvRrFmzqFy5crz33
x3XbbxddffFx0tWrSIxYsXZztiRuPGjWPixIs2bN1N2dO/ZwwwZQUEX/17UrVs3br311tQc6ngptzpzzb49OxY
vXhzVqlWLiIgff/wxKlSoF/a/tZlikNGD2ZLqPyU3bY4BgZ61Kf8vg==

jV69ecf7558fo0aNjr732iry8vHj77bfj3//+d+YkiyNGjIi2bdtmOelPbrrppnj00Uejb9++BdYA3XnnneOOO+7
IuXLz1FNPjTJlysSQIUNSs37l7Nmz48svyywPEuufx5h07PmZjF76qmn4s4774yPP/44IiKaN28eF110UZx00kl
ZTrZ2n3zySYwZMyYzg3PFihVx4IEHRtu2beOiiy7KdryMNd8wfvPNN7HNNttEyZIlM9eVKVMmGjVqFDfccEPsvff
e2Yq4VqNGjYpjjjkmGjduHJ9++mm0bNkyvv7660iSJPbcffOPPPNN7HNNttEyZIlM9eVKVMmGjVqFDfccEPsvff
e2Yq4VqNGjYpjjjkmGjduHJ9++mm0bNkyvv7660iSJPbcffec+9AfEVGiRIlChVZExD777BOPPPJITn4wdsaQmn
xzTffbPDYbbfddhMm+XV23HHH6N27d5xyyikF89tprY86cOdG/f/9sRyxkyZZIl8cQTT8SkSNi9erVsfu8e
pp54a5cuXz3a0a0tbrwwgtj4MCBceSRRxZ5sqm7boHtufzGuFu3bvHwwH+/bti/w
AkitrLKb59Toi4uijj45p06bFww8/HK1bt468vLz8/HK1bt468vLzYOHFinH322dGgQYYP1nh15c7nzzzjsL/Lxmbexhw4bF5ZdfHld
ddVWWkq1du3btIyMhO/dZIrdYj2vrnFu+9916ULL06/vznP8wddFF10qx/9zv3w0F8IysNBvd/cUxqXZlCCW9+s2
vHXfcESVLloweFXrE6NGj48CHaryo3q/8q2l+ttuRokSJaNX3p4fN6JPEcmc4V8sZ6tcgc49Zr4lzV
P6TQ+N954443o379/fPrpp9G8efNo1KhRDDoCAMl72vwNMT7NI/Y5fyTYMb3EpXdaVtyhNKvBHbQkbJ4Na4C3yXkkMQWd
gQoUK8e6776Zq/coWLVpE8+bNN44orrijy/VAufh5hM8jGsfDkjr7LyD5pEiCHzHZdddGqFe/cTd8ZhP7ZYs7jJrIBmyMgoUPrHx8Y/TYzRENiUsBuwwlTkEeCU3lQsLj3D9dlQVvEYl7ZZtR+1wRjPBc8/n
gQoUK8e6776Zq/coWLVpE8+bNN44orrijy/VAufh5hM8jGsfDkjrp16yY1atRITjzxxKR//n+1I69WuXbt

kzpw52Y6xUfbcc8+kV69eSZL83xp6CxYsSI455pjk3nvvzXK6on399dcFLtOmTcv5dWPStoZQkiTJypUrk9tuuy3
Zc889k7p16ybVq1cvcMkly5cvTxo3bpxZDzItypcvn3z99ddJkiRJ7dq1M+ssfvbZZ0mNGjWyGa2QtD7GSZIkNWv
WTF5++eVsx9hge+65Z9KmTZvkiSeeSEaPHp2MGTOmwCUXVapUKXnppZeyHWODpfH1OkmSZNasWcnhhx+e5OXlJWX
KlEnKlCmTlChRIjn88MOLXEM01/Tv3z/p0qVLtmMUqWrVqpm1TKtWrZp89NFHSZIkyfjx45NmzZplM9p6VahQIfn
444+zHeNX++abb5Jnn302VWsNT5w4MenSpUtSqlSpnF2nN83P6TXS+NzIZeXKlcu87/z52ukffvhhZo3QXHLAAQe
kbv3KChUqFLkOK79vDkv/natXr158/PHHMW3atJg2bVrMmDEjGjduXOAQslwzevTobEfYaB9//HHmcNJSpUrFkiV
LolKlSnHDDTfEsccem3Pr26xYsSK6dOkS//jHP2KHHXbIdpwNlrY1hCJ+OhPrQw89FJdcckn06tUrrrnmmvj666/
jX//6V84tZ1G6dOlYtmxZKg5X+bl69erFDz/8ENtuu21su+22MX78+Nh1111j6tSpa10TMFvS+hhH/DQjr0mTJtm
OscE++OCD1M1UqFGjRmy//fbZjrHB0vh6HfHT2tL//ve/47PPPotPPvkkkiSJ5s2bp+b18PDDD4+rr746Z2by/lz
p0qUzf9/q1q0b06ZNi+bNm0fVqlVj2rRpWU63bi1atIjvv/8+2zE22KBBg+Lkk0+OsmXLbNgwli9fHoM
GDYrTTz89ywkLe/fdd2PMmDExZsyYeP3112PBggWx8667xkUXXRTt27fPdrwipfk5vcaa50YuW716dXzxxRdFLnt
z4IEHZilV0Xbaaad4/fXXC80efPrpp6NVq1ZZSrV23bt3j4suuiguv/zy1Kynf9BBB8V7772XqvedbHrKzWK0atW
quOOOO9Z60oo1RUsumTx5cvz444/x2muvxdixY6NXr17x4Ycfxi677BLt27ePW265JdsRi5S29SsrVqwYy5Yti4i
I+vXrx5dffhk77bRTRERORvlEuXbp0fPDBB6krWNK2hlBExGOPPRYPPvhgHHnkkXH99dfHKaecEttvv33ssssuMX7
8+Jxbq6l79+5x6623xkMPPRSlSqXjJeSggw6KF198MXbfffc488wz4+KLL45nnnkmJk6cmJOHi6XxMY6IuPTSS+P
OO++M/v37p+Jvxx577BHTp09PVbl53XXXRe/evWPAgAFRoUKFbMfZIGl7vf65HXbYITWF5s8988wzmZNs5JpWrVr
FxIkTY4cddoj27dvHHtddeG99//33385//jJ133jnb8QqZP39+5r9vvfXWuOKKK6JPnz5FgC5tk5v165d47DDDos
6deoU2L5gwYLo2rVrTpabe+65Z7Rq1Sratm0bZ599dhx44IE599dhx44IE597j+Utqe02uMGjVqrWtkP/LII1lKVbTx48dHp06
d4ptvvin0pXQuLiPTu3fvOO200+J///vf577rn49NNPY9CgQfHSSy9lO14hJ598ckSkaz39o48+Oi6++OK6
YMmVKkX+PjznmmCwlI5usuVmMrr322nXOwMq1kuKX5syZE2PGjInnnn38+hgwZEqtXr87JP2ZpXL/yuOOOiyOPPDL
OPvvsuOKKK2Lo0KHRpUuXeO6556J69eoxcuTPjznnn38+hgwZEqtXr87JP2ZpXL/yuOOOiyOPPDLOPvvsuOKKK2Lo
LJk2axHnnnZdzZ21O02P8y3L41VdfjRo1asROO+2Usye7WePpp5+O6667LlUzFVq1ahVffvllXfeyVKyoqXx9XqNNJSyrVq1KvBFQpIkkZ+fH7Nnz4577wYIF0b59+5g9e3Z07w58/d4wIA
BObdu5Zr1x9dY82H/53K1AChRokR89913Ubt27QLb33vvvWjfvn1OTrqYP39+zpeZv5S253TET0cN3XDDDbHHHns
UuX7s0KFDs5SsaLvttlvssMMOcf311xeZt2rVqllKtnavvPJK9OnTp8EKWd/a+rm4fmWJEiX
Wel0u/j1m81BuFqPtt98+7rrrjjyyCOjcuXKMXny5My28ePHx5AhQ7IdsZChQ4dmDv/48MMPo2bNnHAAQdEu3b
ton379pnZhblkr732isMOOyxuuOGGzCLNderUiVNPPTUOO+ywnDz8+KuvvoFCxfGLrvsEosXL47LLrss88bnjjv
uyMkXje7du8egQYOiSZMmscceexQGGzCLNderUiVNPPTUOO+ywnDz8+KuvvoFCxfGLrvsEosXL47LLrss88bnjjv
cddVV8eSTT0b37t1qj1qxZ2Y5YwM/PSliUXDwMMm3S9BivL+vP5VLuiKLfGOf6TIXrr79+ndf37t17MyXZMGl8vY5
ITyn7y+dDiRIlonbt2tGuXbucPOlfGo0dO3aDx+bKmZrXlN7vvfde7LTTTgWOAFi1alVmnTo1DjvssHjzzTf
dJk2aFB9++GHUrFkTBpnKqTtxIVpfL2OSM9JAPfZZ5+oU6dOPPoo1G9evWIiJg7d2506dIlZs2

aFW+99VaWExZWuXLleP3112O33XbLdpSNVq5cufjkk0+iUaNGBbZ//fXX0bx581iyZEl2gq1F2r4MiVj78+Pdd9+
Ntm3bFjh7fa6oWLFivPLKK7H//vsX2P7666/HYYcdFosWLcpSsrVb19JTEbl39FbalpGpWLFiTJo0KTVrH9eqVSt
effXV2GWXXaJq1arx9ttvR7NmzeLVV1+NSy+9NN59991sRyykXbt2scMOO8R9990X1apVi/feey9Kly4df/7zn+O
iiy4qdMJLficSftcuuOCCpHnz5snTTz5snTTz+dlC9fPnnnkkUeSG2+8Mdlmm22SwYMHZzveFqNixYrJ1KlTkyRJkqOPPjq
55ZZbkiRJkm+++SYpV65cFpOtXc+ePQtcunXrluy3335J1apVkx49emQ7W2gz
98vLykhIlSiR5eXkFLmu2lShRIjnwwAOTOXPmZDtqkiRJ0qxZs2TIkCFJkvz0eH/55ZdJkiRJr169km7dumUz2lo
df/zxyZNPPpntGBusZs2ayejRowttf/XVV5NatWpt/kBBbqKOOOio59thjk1mzZiWVKlVKVPvroo+T1119P9tprr+S
1117LdrwibbfddslLL72UJMlP//6++OKLJEmS5M4770xOeWUbEbbonz55ZfJe++9lyRJkixatCg5//zzk133jk
5/vjjk6+//jrL6f5PuXLlkg8++KDQ9ilTpuTse4vmzZsn77zzZsn77zzTrZj/CoNGjRInn/++ULb//WvfyVbb711FhKtW69
evZKtttoque2225Jy5colN954Y3LmmWcnNWvWTO68885sxyvSMcckxx44IHJ//73v8y2TNmJG3btk2OO+64LCZ
buwYNGiTvv/9+oe3vvfdeTj4v2PT22GOP5PPXXXX892jA1WrVq1zPv57bbbLnn11VeTJEmSL774Iilfvnw2o61V1ap
Vk08++STz3x999FGSJEkyfvz4pFmzFmzZtmMRhYpN4vZJ598knTr1i056KCDkoMPPPjjjp1q1b5h9eLmrQoEHmQ3TlypU
zxdCgQYOSww8/PIvJ1q5Lly7Jy7JyJEjk9WrV2c7ygbba6+6bXXknLlyiWTJ09OkiRJ3nrrrdS98endu3d
y6aWXZjtGkTp16pT07du30Pbbbbrst+dOf/pSFROs3cuTIZO+9vvnlZNy/dfnvnlZNy
4cclOO+2UnHHGGdmOmiRJkpQvXz7z4b6dqZ5/Jnn32W1KhRI5vR1uqhhx5KGjZsmPTu3t55plnkueff77
UL18+8ybt5z744IOkoUKWUi04ebOnZs8+OCDyVXXXXZX88MMPSZIkyaRJk5IZM2ZzkOVlhNWvWzBRYVapUybxWjxo
1Ktltt92yGW2tKlSokHzzzTdJkjh6yWGaymZhUTr98orryQdO3bMfOGbJpdffnmy7x6NiffF6Xxxy5Bp06Y
h7aNGjjUpatmyZhUTr98orryQdO3bMfOGbJpdffnmy7bbbJq+++mqycuXKXKlcmoUaOSbbfdNiffF6Xxy5Bp06Y
lrVq1SkqXLp1st912yfbbb5+ULl062X333ZPp06dn6dnO16R/vGPfySHHHI8u2332a2zZw5M+nYSWNy//33ZzHnX
t2jWZP39+oe0LFy5MunbtmovE6/fFF18kqbnxvLnSyZ5MunbtmovE6/fFF18kqbnxvLnSyZ5MunbtmovE6/fFF18k
/X+C6efPmZTTtqIfvvv38ydOjQJEmS5JRTTkkkO0O+ywZNy4ccnpp5+e7LTTTtkNtxa1atVKPv300Xhg0
bliRJknz88cc5W8iy6Sk3i9HTTz+dlCpVKtlnn32++OLn44ouTNm3bZ0+eufJYmKTNFxdZbb53
897//TZIkSb766ucnel29NFHJ2XLk3q16fXHLJJcm774776b7UjrNXr06KRatpWpJiRIlCrxpuPrqa5Pji8
23ueff55Ur1492zGKdOONNyZVq1ZNHfrNkvbNq1ZNjjjiiOTGG29uMbrzxxuTII49MqlWr9e+i7+oXWlxrJnXfembnki122il54403Cm0
fN25c0qJFiyRJkmTEiBGJNHfrNkvbNq1ZNjjjiiOTGG29uMbrzxxuTII49MqlWr9e+i7+oXWlxrJnXfembnkip122il54403Cm0
2ePHi5I9//GNy8MEHZzHZur333ntJ7dq1kyZNmiSlSpXKzAL4f//v/vsvaZCjvssEMyfvz4JEl++jB
y8803J0mSJE888URSu3btbEYr0qeffprsv//+SYkJQpccvXf3i9NmDAhGTRoUPLPfPw4zmThxYrbjFPLyyyyy8nO+2
0U/L0008n06dPT6ZPn548+SSHp7ZPn548+SHp7ZPn548+Sfqkn1n6tUzlzJlyiQlSpRIKlWqQVGB7rr6GrLFs2bLkpjN
OSvLy8pLSpUsunUXTkqWLJl07do1WbZsWdZVNZSWbbjFZLWL0OSJEmGDx+e3HXXXcmxCtlt92SVq1aZS6
VKlVKSpcunWy//KqY2+WVcuvX7gnRsB+PtndZ6sX+6+Uc11pkWt0+Lk54Uml78
9eyZ77bVXUrZs2WTMI6FXTejZfLvEp5vk+nWuMGn37YtbbZTNcN+QKXwU9nSyXPsYyuTiOyQ/OP7Pb
YY0mSJMm5556b7LXXXsngwYOTQw89nNlrr72yn5scUKhYnTFFVfE1VdfHTfccEB7b1794rr7yy0NkVc8F2220
XX3/9dWy77bbbRokWLeOqp2KvvfaKF198V64YUX4scff4ynnnoqhgwZEv369vgwXP//85+jUqVO
htZByQbt27eL777+P+fPFiog455xMLFillMtvHeeuutnD3z5sMPPxzVq1Pj76KD766KM9mrVqsXDDz+
c+TkvLy9n1qb78ssvivizxxV5UqVTJrsZ2jRnTZx1ZKJXmWJE+HxYRfHM888kmlRrdWHQuebuK3xe+Y9xmk
nLfr16xeHH354bLPNNrGXl5etGXl5eTJ48OcqVKxc33nhjjJZs8WVeyPAGp0dkjkhndFgTg2XSVOVn/U
tZcuW8f7778778d2220Xe++92TP1vzfN8Q0TQDGi6wYCwtp7HCvRPxDrThvGjo3PUWlxVLGxgf+yvGvjqMYYlvL
uG48//ng0aNAguwH/1QporuwNyADRHbXMnxZ+b8cUNv/EH9tRe1XD5HZLMFWK0IzbZbAuUu/9enSRIdX/RnkdSOcjvJ3I
+fPnYeeedc269vzW2WabDlzZjRs2DCaNGKSw4cPjPYx6lo0CFY4kOUZJIj9tR/2nNOyh5f8iObLh7Q7qZ/z8xHW7rjjjjst2hF9
l/vz5kfw0uSkWLFhQ4P38qWr4t///nfUqVMniwmLdtVVV8XFX+1Ht21H0zt+t1XXaAaZCncbAIt/Dd2HGJa0cceLkQxk
ttouPPvoo5syEE9WrV8/Z1/0MfTd5HfC7o8mV9tFb1z3m0n+ivjHBLBiVY5Hz9eIqEcjAFeLPwo8+i+pmhpAEBvC5eVUbcLpsH
NCmz//PPPY9ddd43FixdaWzyaAOpOHB9HoC8M6S2jRj+8t+UaXeUpJ79W2s7z8PRThetaWGAqdgAWSs5bhnZFMdb8eqMnEyxH
5JD7//PNYuXXJltiMVctBB8VzzB1XqDCeP39+H/HHcccTl5ZxPJJksKNnWL0vVdeFbuZjH7Jt0Oxa1wv7OBh7d///
YNGhQ1K5dOyIiZs+eHaffnosWrqXLnvhXQgjhg26M8pN9xxZGqcPnVyXnwX5S55z2DCAvxJiKEvrNyZCfnnzYOSRYVAF
ZMllOuGVYsmRDJb48OD755JNIkiRatGgRp56apQXzXTbaqaqatWq8c4748T2zrRTj8kgmI2XJB9Sr76ygLdFkTsHzRcy
Zs2axdOnSbEcs4JVXXSxFBfFFxFCSecEF999VUcddRR8knno0TN4jXjySfjIMMo0kgOEdfrdv/99b7zzxhsd5e+LCtJ1
UYY2OHTvG/Pnz49FHHHF82chfzTTz+NM844IypWbBjDhwUyFLuUfUMZPXc/pQkUK7g0U9M7doPhH9tm3bdhMmKR5LlizJ6b6
qq66KKlWqxF/+8pd45JfEmGWRMGN8iQx8fBuvCQ84Y2ppbtiQGoG2DoaR8WvZHKWnYmtCSNnF+fFpWPYHmFzY8FrH2lq1D+nR
vvz6uueaazZHq/cqVKxdTWDTMXFHVGqxxDerRGRCUmtiN77OZHi+vc/wZvZ8xGWppPgOmHV6ZYR6H7TKl5WVKMyh0FFdhWeqVJuC
//fVC5ea4cePigAMOyqqgdfv5TZTI27dvHJ598ewbbgtJErYSF7pn+t5qeu/Hhdu1+WVMxl32nvMaqDgsgYnHJfhZjiMtQzUo7qbo9
OVKQxY8YUOktlRMTSpUvj9ddBz0uKBHNmjWWXLTG2264TXpWHszd1EHJjlEQsNpn3IuTxrewaNw7RPRkUsWnZIHsIUaFHCO3auK6Ug7
aNIi8vLyYYNm1abLfddvH8HRMTChQujV69eS6eNbCeC6oxVdw59eFvvjgL2x6oM5oQoG9aaltzaluWRgaxkFCDYx8U54GnZmmxPpFUF
77LMLbF+5cmW89tpr7ceCBB2Yp2bbqVK1Ue5yDpPZfvrpp5ebkaiCihUr4pzWbltOwcmDzdevrpEu8y2Bf1WAMF0Aqxp7YvkEqa8nhGWzx8e8V01tl0D6
pMEqUJdVISrYoSpbRWLIkItuMrbGFwdimyX6PLcsYE411mCFrJNsQBqlkChNJWcpaYkYxJUuFlvP7w6/zdZwTGen

9OfV8XNd9XTqfc3+v5+2bcz6f1/v9fr1atWqFVq1aCU5WOGtra8nsOn8ff/75J06fPq0obAJAgwYNsGrVKjg6Ogp
MpkwTCpZvGj9+PNasWaPyemZmJlxdXXHy5MmSD/Uetm7dip9++gk3btwAANSvXx9Tp07FkCFDBCdT9Xrx8osvvoC
ZmZmkF0MAYP78+fD29oaDg4NG7fYu8OzZM5V7DHWnc0QJCwuDXC5Hx44ds4dsXfvXlStWlVxrXz58jA3N4epqanAhOr
VqFEDFy9eVCluXrx4UZI7TTdt2oRKlSqpnNrcvXs3srKyMGzYMEHJVOno6o6MDc3Fzo7n6i4sLi5gc6ePCg4s9ubm6
YPn06oqOj0bp1awDA2bNnsXv3bsyfP19UxLdKTk5GzZo1FcdTateujdq1ayM/Px/JycqmoXbu24ITqhYWFYceOHdi
7dy/y8vLQt29fBAUFSW6nTVxcnOLV69eRUpKiuLnvLw8HD16FJ9++qmIaO+kqVv6NenBA3j1wBwfH49jx47h+vX
rkMvlaNiwIbp06+NPrv9Ovk8lk+OSTT1C7dm3JHXfTtIclJycn3L9/X+WG/fHjx3BycFG5cbpLsDWivXr3g7e2
NwMBAAFAU6mfMmIF+/oJTlc0rz/oSU25cuWwf/9+ySQb1vbEkpVdMePH8f333+yStEb/1a5dGzZk5csyePRs
dGzk5OSqv5+bmCv/OjouLQ5MmTaClpVVXo53EBW1vbEkpVdMePH8f333+PhQsXKl7LzMyEi4uLwFRFs3z5csyePRs
TJkyAo6Mj5HI5IiMjXbsWPz777+Saw0RFxen9Dvw+mLI77//Lql7iwLr1q3D5s2bJXvPPkOu0
glEJLiDcVLIgkJibCzMxMabFaykaPHo2vv/4at2/fRps2bSCTyRKREYEff/wRXl5eouOpWLJkCebOpWLJkCdatW6fyupGRE
++mtJFTcB4Pvvv8fMmTOxfft2Sd8HvS41NRVTpkxRbFx48zCmlSPwwqFx9SvaBdPPVNQvBal9uRXQ0tJCo0aNcPDgQdS
tW1fxempqKkxNTSWZuVatWkhLS0PXrl3h7u6u6SrYH5OvHP9T9U6tYsSJWrVoFDw+Pko5WKhX24LFmzRosXL
Qcg8emuhdR5rKlSuHAQMGYP369ZL5d2liYoKklS5dqzMOSlpYWUlNTVXY7Xr9+HQ4ODmp3R0rBkydP0L17d1y5cgV
Pnz6FqakpUlJS8NInn+HIkSSOS6C/8Pn1h9+3b9xGT/DcjRoyAjY0NvvvuO2PjsSJmD59VvvvuO9FRCvXmFRjh1QobZ8tXxeZIDLKoDPJbOBW+jeQ9YPfT
FWLNmDZo3bw6ZTIb589j4sSJmD59utCikJaWFlJSUmOEJZ0/g6IS6vVtaZ0oqZwxOVEYCwsLZ3FmFIZffFRg//79MD59OqZOnYq
VOjo6+OOPPVzYxOVEYCwsLzJ8/HH0OHDlV6fcuWLZg3bx4SExMFJVPPxMQEkZGRKj2E0E9+7dq2h9IzXVqlVDVFSU0jO
J1LVp0wYAMHnyZNSscrKOO6yzsrKQnJycrcPagsjckP//8M5YtW4Z//vWJ7kkHAGBqaoqpU6di0qR/uS
TT5CQkKAyByIpKQmNyNGjVCdna2mGCFsLOzw82bkN5GTkwNzc3OVz+26ITk5GRMmTafXr16aFHgAQExOD7t27IysrC5mah
I3Ln5gTRtUIU6jRo1QsuWLREYGIhOnTopXpdq3XvkQrnDn48ssvlQbzSFViYiLkcjksKS0RFRWlVKwmoX748K2
tLTChsvc5jpmenv6R07y/VatW4ZfflF68MefPmSba4mZmHifDwcLU3mFmSba4fZfZifDwcLU3mFIZfFRg//79mD59
WLVtCLpfj3LlzWLZsGebOnYvc3FzMmDED33//PXx8fETHBy/Jyk/3d
UrlwZERERCAsLQ3R0NPLz82FvbH4/OnTuLjqbwZrsNTWNlZYUFCxbg9OnTaN68ucNDiBQ+LzR1qMKb33+ZmZlo1aq
Vor9wbm4udHRO4OHhIbS4mZiYKFoxaWgICCECFChVw+PBhSRc2AeD+/ftqe/na/aMFP84PbtGm
D+/fvC0j0duPGjhMVCYSgaGPxd8Ho2qQ/A7118XFxSE6Ol1qpjYXUPXz4ECN
GjMAff/yh8Zk2RkKRi4uJUipuxsbGGoVq2amFBvIcVd3O8SERGBP//8E82aNRMdHIAS
Exc0yTiaTYe3atfD394erqyuWLl2qeDiS2ipYga+//lp0hCHgAQExOD7t27IysrC5mZmah
atSr+/fdf6OrqwsjISBLFitctWtri1q1amH27NmIioqCnp4+vtJ
gjfLly6N169YqfTilIj8j8/H5s3b8a+ffuQlJQEmUwGCwsLGBsbq925J4qmtsosGHDBhgYYGCA6OhrR0dFK12tymQ
+L6S8S+ltNOX77/XJ3FKd0v0uTZo0waFDh9C9e3c9oaFWVggMDMSsWbOUt+1a5dKv30pmDNnDtL
S0tC5c2f8+eefOHr0KEaNGoVt27Z2ZJtlXI8+fP8euvv+LEiRowWtbVVaWWxfPlyQckV9y9JFC9y9e1ejipvstHj1
6hLNnz8LJyQn79+9HamoqFi5yiGXLlomO91ZSlWmojO91ZSLmoWGDhwICZNmoQTOiTUzM5P
sRiwSh8fSi5km7cACl83/fHHHHxHg0aBC++ILzLiDzBxYWFpJbuStw57N69W3fs1SOEh48eBDdunVDuVDuXLllHq
zqiPVxu6apkmTJhg8eKvLp0SVCywnXo0AH169fHL7/8AgDMzpvquxML46quvMHny5PMy5Pc6Rls
SKlasiJiYGJXpkzJwJCbCzs0N2djZ2J0JrKWsQSmVTZ48GVu25RzRvWtra1GPCzNnz8GVu3boWtra1GPCzNnz8fU6ZMkfwupgJyuRw9e/b
EkSNH0LRpUzRo0EBcvjHlzxs8eG7c9RbpB6NTRkbwRwhIBZhNfqOLBSbxJe0Jv2Opm5wu0pOXLrDHpTWbbq4Jj
8scDevXsxYMAAdO7cjYM6cdGeoAdymJdn8jOymUTMGTa2kOpbzD0qzZWrcdSYvq+fydOnMDu3bvRp08f0RGHvJkCP766y/8/ff8qff2LFjh6SPazo5ObZ93tuR3ht+6dQt
jx47FV199hSZNmqjcY0r5iLbyY+ZDtB5Zc0FBNkpjyE0Rk9Eo5Ej6SeSAIZxjCbkQa8khqWjzp4Q99kW5HAUoWXOB/xRh7ctgrSCiSWdBRz0+ut9bSe1MCcIxRkxbvdbj3hd9hM
Zgt27dyt2/+fn52Po0KFYt24dypcvLzih5jt+/DiWWLV4Kz4QbBOtyk1nQWT66aZm0/Oa8AsqM7D1c/uqZ6uZ58cLd2Lrz/R/gyV95Zr0Bn5ay7sTUo3g96mcAAfQrK5CffDQumw5sbq6SLl2J7jVsCj5JB3H1bsGdPbBE22pEXHn7Ev1cUlXZ0cPrMRbzQLMw3yM6yCRyxfFEO8cMj6nvCTnMCc3ofOgHCtjkLsgrknMPavVXZkjUb5Xb3fwnwOxJ0anqz45xeLljLhMKnCzJUKGfkn5eT0sbQNsqQeTPdG5zKVYpxN1lm3TwsHI5Mm
Y2FjExMQgMjISX33lWSOTL7tYUkmkyE0NLQE05Q+mzZwuTJk3HgwAGVv+vQ0FD07t0bq1evVulTJ5qFhcVH5y
k+J2taTTt6KM6Uv7+u337Nvr06YNLly4p9d4s+L2wvt/v+wzVlOIh8fH0yZMgUAEB0dDV9fX8THx0Mul8Pa2hp
ff/01pk2bhrNnzwpOCrWL6Tk5OfD09IS7s7NSwVsqxW9Nd/bsWQWBhJSUmK1wr+PUr1Hr9y5cqKY9N16tSBv78
/HB0dkZiYiMaNG0ticVrTPzeAVt3SY2NjUbFiRdj0jSBbkRdjY2Eh2t/27+uL8XfY0NAQWVlZyM3Nha6ursqigh
pxcjT0xM9e/ZU7MA6e/as4g9Krfr3S6pG1tTWioQLQp08fyW7x4Xrx4Mfj5+tjxYoVsLCwCwwJgxYxQ
9haTg9aPomnIs/XWZmZmYPn06AgMDkZaWpnDil90fr1w19/QVfX1/8/vvvvvigePqKgo2NniY6nVrly5RQ3FDV
r1kRycjIaNWqEklWqIDk5WXA6B6VVvWrIGbmxtq1aoFW1tbyMXF1FIS8vD4AcOHXG6oH7m2++NmvsR0r0Nm
zB4GBgWp3Y0lth8LOnTsxa9tcfPbb79V+jV+jkJwcMTE4evoPpbk6dKkibUOxQ2SEgmk+G
TTz6BlZUVevXqXqJZlpp5p69FFTvv8mT54MCwsLnDhxAHbOy2Dh4/o0tTWHrVrOy0tbY2bY/v27Yp
rBQ0ORpDLo7W298/z8/ODn5wdMzPVvkHYsdLKnaImzFQqfcsmJmEm6qRWxMx/j7d69WpMmMm4eHB+zs7
DZs2aKXa/rVu3TJPUr+uQEA9evXR/369R+wNM41XGFbYrxT6+cr/VxwP7Rly5lb3KjKXJF9fX434t0Ylizs3i5Gm7cDSVHp
6erhy5Qrq1KmtWaLmD6tWrIywdDY2l11sxJycHs7OWL+LvXxL9+vUZP348e+vUZ8wRUYP348wsLC4O3tjWHDhgXSMsw+++
/sX79eixZsgU7YKzs7OGD58OAYOHJg2fqoxyxY8ciJiYGkyZNwS2X3ZSbGSI0xf6roOqePbsGbZvZvZ347r169DLpe

jYcOGGDx4sOT7IN28eRO3bt1Cu3btULFiRUn1gnzdypUr8b///Q/Dhg3Db7/9hhEjRuDWrVs4d+4cxo8fj0WLFom
OqMTY2BhHjx4ttKl7TEwMunXrhpSUlJIN9h+tWbMG58+fl8wpgNc5OTnhwoULyMvLQ4MGDSCXy3Hjxg1oa2ujYcO
GuHbtmmLHurW1tei4GnH0UR1N+f6rXr06QkNDYWtriypVqiAqKgoNGjRAaGgovLy8EBMTIzpioaKjoxEfHw+ZTAZ
ra2vJLkACrxabhgwZgp07dyoVDzMzM+Hs7Iy0tDScPHkSxsbG4kKWEgYYGBggODkaLFi2UXv/5558xZ84cyRSRX6e
np4fY2FhJ9l0tjL+/P3JycjB8+HDExMSga9euzpSa4dUmB07dmDXrl04cOC
A6ChERcKdm8VIU3ZgPXnyRHGk6l03CVI5evW6qlWrKiblffrpp7h8+JsbGyQkZEhiWMUbYGBggODkaLFi2UXv/5558xZ84cyRSRX6e
FBWHr1q3o0KEDPDw80LZtW1hZWcHc3Bz+/v6Sebh7U35+Pm7evIkHDx6o7JgtaOotJYsXL9b8Pi9YsADDDhg3DuHH
jYGVlJcniCgBUqlQJY8eOFR2jyNLS0tC/f3+EhYWBBlcBS0xKhRo2BgYCC5HWRr167Fr7/+ikGDBmHLli2
YNm0aLC0tMWfOHEkes0lPT0fNmjULvV6ZzZk08evSoBBN9mG7dumHmzJmS/PdXsCtz06ZNSt/jeNOxeff47Ro0d
j8ODB8PT0xLFjxwSnfVX8KWh7U7VqVTx8169eHjY2N5N5HYgv05Tvv/y8vJQqVIlAK8Knf/88w8aNzXLt
2TXA69R48eICBAwfi5MmTMDAwgFwux+PHj+Hk5ISAgDHp+eKLL5CRkYHBgwfj8OHDcHJykYwRNzk+Di4oKHHDx+
ysFmMfH190b17d4SHhysWaHx8fLBgwQIcPnxYREHFzaysLEydOJZSbShk6NS9EvYksnJEkgUdEOTt36bUlZHvJm/Wd55+x33fJlHh
A7dq1Ub16ddExAbw6dlzUZycp3BsWkCcLd4ObURbVqF7uqm8cG0D7uwRZ6x5OhZfDpyXt5/nnn7RX4ZBMtj//0c3EiHqNAWloajIyMJLkznT4+FjeLkZ2dnWJ
XgpOTE+bMbrkCoaGhooaAwMDA7ePqi9Zf2q3Ph0bDdsPgq9aAUtM/kt1xrpMvRdq4kDg4IcNuTMT5iT6Ihq1dePGMxWvVK3vA4+ZZlIDSZIY
INnToUGzcuBFLliwRHaXI0tPTYWFhAeBVkbvgxu3PxEwaTCr5w7MmzWLLihXoVRhhIRcqVn2dgPEvsk6foTGG/aX0LjvtfQXjq7eWcSdmBxTjw/k9Ujb+TF+Qx6h8qXnaVzGkqVvGtAQ9qHsO9P
EplFP04djeXp6oly5cooFpwIDBgyAp6en5IqbncnJaNOmDYXA5wKit9DhgxB69atsXr1apHxVOTl5SqPa4T25qujra2
N3NzcEkz0Yfbs2S0OmZn8QSF+umnnnxAcHHKy08Fi5cmXMmzcPzs7OmDx5MubmmQNnZ2eBKf+PJhx9VEdTvv+aNGHmCuLg
4WFpaolWrVli6dHK6QQ8efIEV65cnm7fFrGr/6oaoiOXvRfW8UOEyVlFOfZ1d0Ol/pzZqXKUOIp9lXm1MzIIyD9Aw9qKWe6ymVNKGwWgPdqTNFG
7d28cOHAAs2fPRkpKCsLDw2FqaWOIzJCMRRDQ7R2FqQXK8lSLRDtLnuSLY/4pDjodXWIE0tLS4OzsjIiIOzatQuLFy/GH3/8ofhelJqePXv
C09MTly5dgo2NJvyRdNn2GnHCHPbPnHB9NuHuHNBmDcDH3YiXmZzwsRPBXtIdpm3RuZXHRHJDUVQfHlTecj9r31M4h4cb47u9R3bKjvfMtOEDoXL7fHovWxv6j8PbZ2S+XS
VqlWoVauW6Gs7r2+qLSl7cLxyvn1eUOFCuejGcddxkxFLUCbsAfH1XxsK7mcpoUN0zFM9d5DbZmTZBbN79WmUHB0cxyWXr+JT7JNx9lpDdAZ4RNQs8HVYXUg1
Pni04nXovX77Ehg0bEBwcDAcHB5UpaxlRxx1pbQeHIi6QrDyYFjvc0eKKCGBiQjaYaoa0H2yE8MFBFwLKwU6UMFDAC9tELnH4G2Ym6Xq4DHe
+DBMTE41aHZWy3r17K4Zjva2fl1QLyMePH8exY8dUbtDq1auHO3fuCEqlTcBJTRsEuBqxnZ2FzCFgsiK5cjAWC6qP9HpxqrZYh6RvuTH3tYsVhj8hHzeeGsj9iDxyBjcfsPPJyL9gi6SnTfiqtSDqS4tPV8Ys
EREn2QpbL5Rg+fDgqVKig9vqLFy9KOFHRvDmJVS6XIyUlBQ8fPsTatWtsFJivc48eP8eDBA5Uj5j5w8fFlScxDAwMF
p0yrKt99+q2gVM3fuXHTt2hX+/v6Ko49Sp1Snff99//z0yMzMBvLrv7NmzJ9q2bQhtM9q2BYb4hIbc4IC6oDrFhjh2h49ihMnTig
tNFlbW2PNmjWSKcoXZtq0aYqdkf/++y9WrFiBKVOm4OzZs5g9ezbs7e0Vu1GXXbqMVYtqDYXBEhXeTjcb+l9mzxTyy9nXfT9Htq3rEBSS8qXB6NGjWf0UXyqxBwgEDCBWcEBeXh6
OHz+OVq1aiY5VqIKTBvEwi7u8PR0RF5eXnQ1tYWnGHDREcoM97cJSHAtWpOgRqPTEDJDMncaQdh4SRR
06VPo6uoq9RiWgoIFHJlMhg0bWxgd0biopHgrHnJil5/Pgx/P39sWHDBsBsTGxkrqAcnNz4e/
vj65du2rUESBNnNbs6+sLbW1t3J//cvGlyQ7J0sS+R6mpqZ1gyZCQkLleXvVs4MEDleKV1P4
NaiJ9fX1cuHAB9erV1Ml93qVHB9erVg76+PmPaPmJjY2FaPxtcQHR0ecP38effv2haGhISZMmKD7fTz9OZ9+ymAQkUaNG
NG0VHVDJixIgivU9Ki3tRWLS0t1KhRdDmDZcuWIITAGAhcKvsKUKVPQk0bbNu2DQE
BAfDx8cH58+dFWlZUluaOP6mji91+B9PT09zrOWdL09fXx559vqevTojYmJQfv375WNmzcG
VKWzu27evJGOVGoXtNvVxXx8UG7du3QsmVLchqivLlyyMxMVHpd7dixYuq4fv06zMzFzt3e
rHQwptc+KLVu2KP1ccD/UqlUrGBoajit37qBWr7ffny5cujp068Pb2Fe7Of38/LB
v3z6Ym5ujX79+Nev6+6NevyqiI+Ph7m5uOunH5HX+/Xrh7Fjx2JgYDbHR3fbrHfyQWvPnn5+PPv8+rLJNGfSuWzrZoE
DcnIyJkyYoHa3aa9evQvQlKz1cXV1+hb2+PBQsWqwzJ3ENNmaJrZoE6f4/f4/tPeeN1gX1nCPFm/wPxXRIhPr6pG4GEiO2H3I1YpDCgSV8JRhR2V3P8P3qVQV+n5+Yqj3oGBgYi
IiICVlRXGjh3L4zbF5O7du4U+U4s9czbB4QmUtbBVCqPZKeqpeW2bLs3fGTcF0zdE6sgdHrJ56gP79PxHoIxJg3w5tUezAQMzeq7bXfMDqyMnc1TtF
eStKNGzdQr149z+mNS+/94stqmjo6MDY2NjNc5OTnSBGSLSB23tmsUbXdpGPvtGgtx9OWYmQmdK4wbgQhFvVh0ubPEtb/dQgo9ZM
XCjR1Aed1BY97Uix6FxQr3kUmkb4D+H27dsfbo0U3ppq2tjZSUFKXgX93RUX9/0NJeb2nnopaWlka1ZU/QsICMRQoUPh7
M4OBgODs748aNG0hJSUGfPn0kZHZCyRdPB3zB9JLcYLBsy3zoJwlzZ4r0t8F1eI0A3BIN7FTV/fvfc2RIvuyDVwt+LTmRDuyQdlMiDBaJ6fVw4ezZC6N/NMxR+k/dDuMPsm3s+ZbNmT
tW4fY2FhJTFktjJOTEyzSn+fOvWLXz//feYUMhbHFb8u4DDcyOIzdrcOOy3D1xy4b//37Q8h6DaS2Lq1FfLGhbJDaY/WIXxq5h66jdCbrWHeZFsqnx09yKeW3tqSHL
aN2+O0NBQuLm54ePZw5cpq7vS0GB3ROqWRJJqQ80mJ0q0FGt7tEC0KKwxbBvzXaSNbLXZGmLQ3RWa6Cv0cQcQn2jqV70ZSBl4rqfwe2ccHx9WXfbc5gLuq7LccjSG
tW1fpWJaUaGlpwcTEB03bt0f79u3hx5i9S0lRxZMnx8c+vPhYHRxTHiJmtkUkpT4MKXeFw3vXmkysvHmv4dv0+P7AN3+g2WwrWpV1RIOo0Oq8gDkpqqrNNl4DLHyjc
UtUdmRbh79y569eqFy5cvw8zMDDKZDBdNDNyKZZCYhGuI5B+OFxB+uW0sIuD5RZaWbwbFhQE+kqXOVjkh4FpnyKaw2KcANZ7Xbp9Av6+Swv
869/WnDlzSijJu2lpaWlvpOaaOFbtW5KbWSCgoLQsWNMZNHyZZ+ZxZvg0tNH84r8JZqDY7X+PaCzi1AOnRCywmGQP/ehQqzWjVZN7AxSepxVkH
Hj1ecGLKwsMCYMWMnYmKicnJEChMe49eoSZpPgkIZ1hPkXPepmaMdL1Mzr1JBqg3ixZO8ZmnYfHMvDXeAzxY/9OOrnA4cH+2YVbR2Yk3tnBDDc8D
++wzAq90fV65cwTfffIMVsCebV3hBe0RVNV0hVBsd8Oz87uL1CBOMFyW03eWQDDCQZVAJHrJsCbQGWwmsjT6fHY3D8f2wPWhltUwJx5eU
VUixgPX/+HKtWrUJYWJaKd5KSKNwDhRe0ZDIZEhZWaFdu3b/Ea6Ur1qNZinY4AOkCMm00yQNH/3FnZhOQzO66+pOIDc8FWgdKZmu0/WpTsKRCNAT03zWKBVGkMKYLbsg8S1a
8hw8fIisrS6ZBkUMp4t6q4kQNWEHIBT8ssvTAmcAq6Db2UrlBGXryN6Sv0dRPQhr1dkO6uT6/mGY8EWCeAlmWnJnW5Tb7lhhw8UR7DjZ6Pr5+ayqepTOB0zTETsKq
+AODUqVPo2bMn5s2bB09PPT8EJNV9qaipCQ0MRHh6OKydrydE4BvtbGz6biHnN
jkJycLDqKkuDgYCQkJEAuI8Pa2hqdO3cWHanUWb58Obp0JEyb5XHNcq8rkyWFoEJtFA3dOv3cbOwwpPwJxgtJyMVwQYEeGPM5ZiRNQ0xGWBVJKtRoe
OEhMToaOjg7p160rOqHr0I7ELWBp6erhy5Qrq1KmD6tWrIywsDDw2NoijPpPpMm0rr33lhWzEZsspJuxBE63AG2bB97DHOnC6DYwPlXTc4u0nqs4k0wNOcj5UvfGMD12jJpWodIQ0D5RNyv5OlPtIHa1xk0H1sS0rZ0mlhJbGKV6FL
0dDQyMjJw8OBByRWPC9y7dw8HDx5UeT/RfivMs6ONjcbMYjRo1CiYmJliwYIS63PnzsXdu3fh5+cnJkqqHR0dTJo
0CePGjVO6edewE4qaWlpbKawXFLakWsAYPHozg4Gt4GB88cUXq37lz5wpKQM0BBByuVxRKpUqRI

ePHgAS0tLhIWFCe3T8z4DYaTYyuD48eNYtmyZYoqw1O3YsQNr167Fxo0bFTuwrl27htGjR2PMmDFwd3cXnFDzhYe
Ho1evXqhcubJG3WRqGrlcji+//BIPHjzA8ePHcebMGbi5uWHhwoWS7ano5OT01tMiUuzf/LqbN29i4cKF8Pf3R35
+viS/rwHAy8sL5cqVw5IlS0RH+WAZGRnw8PCQxE4sKlkWFhaYP38+hg4dqvT6li1bMG/ePCQmJgpKVrgvvvgCDg4
OmDFjhtLrP/30E6KiorB7925Byd7PkydPMHz4cPTp04e7ZItJRkYGoqKi1G4SefN3XDQzMzMcOXIENjY2aNq0KWb
MmIFBgwbhzJkzcHFxwePHj0VHVNKkSRO0adMGv/zyi2LTSl5eHr755htERkbi8uXLghOqCgkJgZubGywsLHDt2jU
0adIESUlJkMvlsLe3l/z9EH0cLG4WoypVquD8fMqK/03btyAg4ODpD7Izpw5Az8/PwQGBqJhw4YYMmQIBgwYAFN
TU8kXN99VzJJiAatKlSo4cuQIHB0dRUcpsp07QIB0dRUcpsp07d+LXX3/Fhg0bFMd2b968iTFjxuDrr7+Go6MjY8n0LEx
LS1McMb179y5+++03ZGdnw83NDW3bthWcTj1DQ0NkZWUhNzcXurq6Kkfp27dutizZ4/KToXo6Gh88cU
XknxYAjTrplgTbzI1VU5ODlxdXZGZmYmYm4uDj88MMPmDBhguhYhXpzp1VOTg4uXryIy5cvvY9iwYVixYoWgZOo9e/Y
MERER0HnyJMLDw3Hx4kU0atQIHTp0QPv27SXbUxirKys4OgoHJChDtCPkxISAh8fX0VO9MbNmyIb7/
9lrs3i9knn3yCy5cvqwxZvHHjBmxPD8+XNByQpUnrKys4ODgoHJChDtCPkxISAh8fX0VO9MbNmyIb7/
SkugoGi8oKAju7u7IzMyEvr6+0iKfTCaTzL2yh4cHVqxYgTFjxsDBwQHfffcdFi1ahBUrVqBXr14IDg6Gvb295Ba
bKlasiIsXL6q0jbl27RqaNWuG7OxsQckK17JlS3NjY/zwww8qW/4I2I5NjY/zwww8qW/4
3bdqEGTNmSPILOSsrCwEBAfDz80NUVBTy8vKwfPlyeHh4ILI7p0YeztrZGQECAJI/MF6Zu3brYu3ev2omm/fr1w+3
bt3H69Gn069dP+PGKS5cuoWfnrh79y7q1auHgIAAuLi4IDMzE1paWWsjMzMSePXk2af1zQmFbxo2bFgJJSkaXV1
dnDx5Uml6KQBERUhQ4cOyMrKEpsscJpyU1xAE28yNcXrPXoLPH36FIMGDYKrq6Ckjx6vSzbAmfV7PmzcPz549g4+Pj+g
oSsqVK4eatiyJAhcHJywefMFeff44qVaqIjVbIBVqIjvVOTk5OhV6TyWTcEfIBVq9eDU9PT3zxdDdM
0adIEgwcPxqxZs5ReX7hwIQICAkS5UFbY919CQgsLs7Ow06vsvIiCPXv2xKNHjtvNi/3DsOUhXj169Xz
OU6iCIUg6Ojp4/vw5TE1NkZ+fDx8fH0dunjmzkTW+YVLTlq8qmk1nq8ts4Pu+80I5SlLgpFqWj5FpcHmfdmnBCNsHlLfSmcL4cL3BjxqL8zZEM2AhrL7akC3O
r1oWhoSEiIiLQuHFjxMbGolevXlxUKKNY3CxGbmhYsHFCI2L
bNmRkZKBLly44ePCg6FiFunp1uqtoeG25uboISFe6PP/7AypUrsW7dOknuL3Dxo06dUhyLLXDu3Dm0b98eT5484bCoEAJ2
ZSEpKQpMmTfDs2h2TNBKV/p1q0bdHR0MH36dL6dUQh2Cs7MzNmzYRzNmzYADDVLpzo6GicPXtWaM7SoGfnkhOTsbGjRv
RvHlzyGQNyHzPwVbkCehOPT2JKh4CdGTHY2gefgXmfBeAy6RwAvCz6Y2hcmb4ObBNcD3Xc6GpYd6Y
iP4OaNRIdjQT59NNPMXmTJUi5po1a7Bo0S8888/gpKVPnv37sWAAQyZQbgtTaTISIiAidOnM/Du3bvRp08
f0RFVtGjRAj179gQZWjNv3jwEBQUhOjpawjLk/Anan6XRoxDvaUxTCCBBK5xjkOuTh4R4OXa4RXvoYWh1B68OP2ay06X
ttLS0kJHSAiMjI9FR3suuXbXbswbdo0TJw4UamGsWbNGixZskeGi8zIW3wE/H4xBpA1gDj+rnHbhbhTtqcdhJL1eHCU6ZN5evcbWf27DrOXHzZyMOY2
Fo6Oj8GdTEoPFzWIWGBiIFStWwECfZ+0Nb6Cvn6QpsgpVDUDX0rXbaBB9CQmJTAhVDQzubWGrYBc/87S7Rb
AJPlA+vDhQ/Tv3x2t7q+lutWT1nhcubLiz2+jq6sLHR2dkohVJA8fPSsFPsSwYcNGjKmpKUZFXNeEhA69atJT
Nha2tLZ49e4bKlSsjKipKUZhNSEhA69atJTP9fMmbLiz2+jq6sLHR2dkohVJA8fPsSwYcNw9OhRxe9xbm4
uunbtb2bN0vyhk4Tbopf31EYHx//1pvMAQMGiAhVPuQEcA6A6hHjVMqagBUDB8Kavv/4
OHRAQECA62lvdvHkt27dQrt27CxYkYgjBgBkhVMqagBUDB8Kavv/4
a06ZNk+SC78GDB9GvXz/XXz8MHjwYHTt2BPCqjcHcHOnTuxuce/duSyZPA8/Z71o/4uWD48BGUuDHB8Kavv
K/nlaS0sLqampqFGjhugo70XdPIvXSHt3fv3nB1dcXo0aMxbdIN+/H8OHD8e+fftgaGiIEydLAjcXbvEA
ikFanX1x5bTBhFtMhH9YPLaOfahM8DBKNDO/aGXoaGkIDqsO4T+kmnKyX2jMzXrNzCZN4VoukQLTNRsh8e0tP
ZMpGzxnmpWFkrWlpKHVEb0pWzmsSbd/z6sy21yf1xFYlgzE0+t6XpYBfFjZBdaKxYpKZbGzjatbB96x+aXzW6sXn
SkZuqs5SP4hcFXTPrnWhYM2cNPpWwVLs5vfyGHawKaVntCWzZ3Ex1tdX0bRHJZdWDpDDQsUM2fT5c6Pq3zAWsMX3s8vsLj5Y
ApbuBKelZnpmz1gUnSHyW2CLJB0G1jpeFvDhfOGQtuFkW7zHVN8gwJF6sGUNoThn2nqm1RCBmn4yKEyOlqzY5z3jkR1G09sD3dxMe
hkLq1GZOZUeW8w7CnZ0Tv4E1GNHZRuVXeOMZKhKjqj0fdvZzBoxCrTyQA3qA0RvRGxtxzeqO93YhKIIwDXIALNHJYcM7b3I2NWGIhazCuFyEfYhwgBJuxY7GDK6ZMLf23R6Mq+ox6Lbx3u4zEWw5m4FAIhUpHQmrjtsQMTHJ8U2XXeG3Y4xQINuuaDaqjZ7A+/9xqqbEq2SmX2lR5v
eYVGQU8IDcXKLHu6KSamMyHe1LrX1+dghZxqFdIkb2dCj5tWWqqL17aHqQCxQBmWfYYDzkLJe1jdF8wbiECM2X7tMD41j7zNBPggL0jLUR6KdlS+UOdEvA/MaUlVR4U6WQ1ejDngkMzwQRd1OkjaR9FUrnU3wWyAP5Nye3emzLT2vHm1m
+/FFHMrA0K6DR3AySKcvtp4cKU+SmXBs5K2uVBWvmLCJ+wJAPsjvjj8ZAkFt7yPFt4+tAsL8z+JzKzR3YJKpNP0X4PpEDbqPKXd4jbI3NgXuZGTxD2Gk6
mBb3QG+cwRrBNe8Z7SGRn2lY1kxMzVWmGuhrNRGCY8s2Q8dXTbG4rBp8TjrRh5AI34h5Qyt5Oz3FApWA01PRrTmGNeZX0ZmNzaKbwJ4Oy1E4hxEIOi1kM9FeaTG7mONkTtlMsI8+qTMYPkjD3iI0HTKpTBW9a6XkwQYg1u0gEcd6HtIUcD4MW6MQvN0TmEgu6r4f+iTc4bOZBPb81bKEcmrNgGZHq0IkCDIETQlRPPUwCxA5DjK1GA0RJN8Yq1eZtNT6ycQcqGR+aRB7xcD5uCTDeMp8F0mvDLLZ
7fRJO4JNNsyeQGSTVG4LM7cXFz9k/ZzHgDG1nQyS+Bbs0A8UYEo8AeNLyD2Ye3KxnYKNSpVNFqhEPZVHM91IiQnCT2Dz3bPiXYOkuKnYHWa7jXHAHXmPUuXiKhgODlGWnXW2ULgS3jPHJRdJSvJlGNYj6ZL98sqgsQ3wr3B5dwm0ovwA
4y6ZBeSEDn4HHj3RJzZhLg0PC0g7iKMa2mPNyO2dKZdCTnF0LNaMPORg/W+tcRwMGOLR5YTmeVEr/XptHHYLh6fQYuK5UncL+XlC4g1ebFg+Ac6rlaRsJ7pwYh+gcHPmNuXfYLlvSPZbHOiSOr6lA3dsHfGdIfZQPbSFT9rABx2Kmt
5X3ZY0IaXJs8ebWvjTDkmfl9wKszbF8SiwM7GpjTMvTEfPgTWxmmSCRLSlDCxcfRv99dCnWXB5W+Tw+XWG+2UZvWx5Kkx0ZMFJ/nW4xUjv2GQ2hbAb2FL3LUdYU1gWF9vR4m3i0hkR8G4aM0rtYZxSxZs27rx5AYFQ4U8ajrFUt6bKE4QYhnQ8fXvB+TkoKxGrPvmFnDnvsxc1ZoT5ZCE9Vc8/LnwKeG
XfB7TWE2RBNL4nfxMXRW0IIKUjkmURxFCEIYwTAfeHDoUxdN1u1fPmk1eOIYA8b5IHgcCc1cT29F7LLBpr2Y1v3wNEsm0kp4t2KzIG1OAhh5rqEgsTgnMBpfWT8Mn7H/xfrd35bmFfmgeSRaaBMF2IkNr6jqHz/o7MDAIv5IZNBFxwWp
xP2eI2QXXEYqHkydP4IdugvwkmYQOmh+7NBF5WHjIDUqxO6/oNFjnoy7kE8eSoFm0Zl0UR5J3XZfESo2/EN/+gTV0vNzLb1Bs5JeD7+Vv9Bf5sUvWdm3vAYfD3cOyqVJYDR3tDZRR7Wa+Ms/XD61xyJfq/oj/UiYnN9MQ/CZNlXKUgwLe0EuC/uPo/HLdd5VMvRv0nEv8CE7bwTH9OYDmX9cl9oD4UbYhzwKDr
dRhc6+SXm4mzXRYRuFC1grq0Z+4j1TKbPOUuS9a2Vbf7Se+V6aX5P4HvjP8Qh8kW2SLaUe2aV6V8kW6V7d2hMFBb7Q1w5PrF3gJ7N0nMkvH1kyVCP2l/DBEPwCl6lsB14zQ6a0zzmz4IuuaZKKo4Wr8tB6zkFzJiRzSy2bH+UA5fK9bKaYyHVmPl9Q8U9O4AyJFIX7Qj9z0Fxo/pFnB3

VqVIFFy5cQN26dZVW+u/cuYMGDRrg+fPnoiOWCseOHUNmZib69u2L27dvo0ePHkhISEC1atWwa9cuRQNyKSnYRfb
ml57UmjO/bsKECYoilomJiUp2X19fQclKl8zMTEyfPh2BgYFIS0tTuS61343q1atj69at6N69u+go7xQaGooJEyb
g7NmzKgsIjx8/Rps2bbBu3TrJ9endtGkTxo4di/Lly6NatWpK//a4m6n4aGKLkxYtWmDJkiXo1KmT6ChFcurUqbd
eb9euXQkleT+urq6wt7fHggULFG1OzM3NMXDgQOTn52PPnj2iIxIVycyZM7F06VLUqVMH4eHhqFWrluhIb3XgwAG
ln3NychATE4MtW7Zg/vz5GDlypKBk6r148QL79QuZ2Dn58fTp8+DVdXV4wcORLOzs4aW+iSEhcXFzg5OWH69OkAgEu
XLsHe3h7Dhw9Ho0aN8NNPP2HMmDGYN2+e2KD/n6bu3NTEeRbOzs548uQuQJtmzZZomiZdu3aNXh4eEBPTw/Hjx8XnJB
EYHGzGNWsWRNHjx6FnZ2dUnHz+PHjGDO7evSs6YqmVnp4OQ0NDyd5IvK2vYkxMDL799tuSC1NEmlTEmTjx49
HWFgYvL29MXToUKxZswzR///031q9fjjyVLlsDd3V1ORCWmpqY4efKkpAcIFXBzc4OTkxM8PT3VXl+5ciXCXCwsIkd2T
MzMwMY8eOxcyZM4t8vJ7enya2ODl+/DimT5+OBQsWoWnWWkdgpA3e/v69/TUlu8KXD16lV06NABzZs3R2h
oKNzc3HDlyhWkp6cjMjISdevWFR1Ro927dw8HDx5Ue/xx+fLLk+Hjty5AiaMm2q0vt/3759JRnrg+zYsQO
7du1SKX5KyKyZ07d7B582Zs3boVOTk5uHr1KipVqiQ6lkYzMTFBUBUFAQHBwcAAD/+9//EB4ejoiICADA7t27MXfuXMk
Ox9IUmrjYW7Fw+fLlgFVEm8OOHjts5AiaNm2q0vt/3759JRnrg+zYsQO7du1SKX5KyKyZ07d7B582Zs3boVOTk5uHr1KipVqiQ6lkYzMTFBUBUFAQHBwcAAD/+9//EB4ejoiICADA7t27MXfuXMk
Ox9IUmrjYW7Fi
Ox9IUmrjYW7FiRZw+fRp2dnZKr1+4cAGOjo7Izs4WlIxE4rHH0YtSrVy94e3sjMXDQwkMb+OTkZMPP9P9+vUTnK
0k/oxkDcnPD5+/Bj+/v7YsGEDYmNjyHJVncLF++PKysrETHKPWCgoKwdetWnHWBVEJ9W8aIQUjQBDGN8Ol/f39nDe36HPZJvP2qGNHnxZymhYBc4tSDiXBWhZ7UVu0sPQ6S06hBagOK8ChFtF/8o/YYkKxDp1IYYhNNNZC1E9sL
3vby8sGLFCqxevVqyiwkFYmNj8eOPPzZkLjoFDKtfOGDjAs0PFdzk2OfmD8JHiYNSUcA0gRTkxIGzktcCaNKjRJhl4Z8xCqjNhNXHBiSkC6Rv4oWkiLvhv7sJFEiFcl0C3eqxNDmDgbu7+Kl1o6hM2
LdmHNnj0bixYtEpTq3aytrREXF4dffvkF2traitMi48ePh4mJieh4GE0kJARubm6wsLDAtWvX0KRJEeul8P
e3l50vFLhzWEmgwYNEpSk+LRq1QqjR48WHeOtZDZKkvprfGkYpHjx6hZs2aip/Dw8PGiMVDFjm4H5zWEWoSyDdsUZ7pTTUbGSlh0vD6mNZCafKHUBaHhy8RTvjyYHiWZJNX1XmqG1s6sbRpXYX0Xxj6emG2LRbb8u6BL1Q1Vqh7TQmCY6QYCabPZhXqwtKRdCP0KIAlD3CLPXO9VvBRYLJvDGDxEsYZ4bI21GKLdi2+s7fL21u8FbRAi7ZM9wsWBYyWsXw+F7tg5nArNpNPfNaR+
F+OrVq+Off/5BgwYNYG5ujmvXrglOp17t2rVVWnkBQG5uLoc4l2EsbhYjHx8f+BhMEEsvu2zZhrfbugwHtWJmG0lXvXx4kbI+cCF6P5w/c9Jqi0yUbBNvSrtoa6ZsZpPtUYcYB/ZyK3VdabaR6nFYMM+WbFRbL+tsN0IMRDb0QZJ3Fq2bJaHWdHWQT2ePvcHgMyMAIMYIgVn3xIYz1Ugt2gKGdHxlY66A7iqDHj0EZQXKjvtEAq9uMNC9K38kpxGJW6CXWhgRAfphsDfScaKNiEHF3KsETjR9bVK8O/u4YcKfSsdL5oGbLmXdzFHfTVXFJz8N1+Vl0WD5t4uA6t4whh+IPQ48fUbDEiqhAdlUxKfSrKGmSQFQ2+ERL1Oa8TBnJdm5MBEOrJ5UbexgoF5S4I0BxlDf3NXsiFiL0blFTHJnp3NPqKL3PGEtsQUAx+8QdNbK0IS0Ys6JKlwdc5GuGmjbSCmPWdnaXhSOIHG4x88SBcEKu8F+AX8gfxrkGhi6bNGDPbqGMHZ7b8xOtV1+jwi8E2l1C0pUc0E5eIAHDzAvWRV44JQTvF9bvN+Z4Ebm0hY47YIsqlQamDSxvhb4PoBSwbQ1dfwBklz0XbkMRyN7Mx5OsF6jHX2t9iDw8ch9nvQOZHCA/SHvNxezIhWQAvsUfPxTEv3XD47ExpvWTVAVb3Y8V77dMSGtT2GgMzMHGg8n7U+s+Nx54tuabWmBqmIz9Z7QLyC7dt+x1nm/5Pu1Aq0nNCnZG0q5+FYSXoNxOWy9bHmqNqSbpEKpHvDXZdqNrcO3LV6Dt8eTVFjTQmMXDXJ7+gzOsYJlDAHTr/IWfBYAwHi8YsL3JLMfJZ6lgHuGVUnkJUJLZ1MTUnqdWmoUYX6CEKwe4Mwp8j3t2X5MX5S4XV3a3NtOIDVIiLLfW3Dd7wGD6Z2mPmzn6mpU4iTHi1Z7A7Nw/TmQtfCA6fT/5eCYHXJTvomjXrvpwZnbO+DFH8o8ZNbdsEWJ8FEkngD2y2Ei7ZgTS5ZiQ+hlfJTFXjeMw4i6a0bSHbmeGrdmaOJM7wvEVA78Wd6qPk0IcB6a5v3c3Tsq13v4Q2N2DKWm8KGmhpzZPkYbwwDLZgptB8oMdCvdEbt5Rt2n2I3KxUzZ6Nl9Dl3TZwP2Jq3X+ARbHuGt4TWW1Sd+Wvv9xjXKiHxmCcydJ1Xr8EyTD6AYNKPWOCL+9IsZoOoTYl3jeabFGUAM8GJEqvKyJxCV9FVWRDbHLXvZZN+P3flVQW/TvfbWLv/0K6cwGqnKSAaH6hnSNGl8XSXO0aHYsx+Y5s3CHNmPTBZGhH/yUI2LCj9ndVqzZf3aRcPBaZFJSj5cWPD8LUDAl8S5sHzUPCp6Hn4jjbYbt9wbIIJX09J3dDBj5N/bxXMyBGlXnbmMO7wVfmvmYrUN5nRnbD7T1ImkhlkJgaeBsRVbzdIYbHObl8sOTP9U6VWzTZiCAq2EpVrGrZkMX7xRKcQUP7w6+KYYdRT1tmuW9KXTLbtqhjc/0B8TkVMvVQDUzlPeaqCSxuHymJ51m5LsqUXTZKs5GUVEqNNUqwE5nLCWYKXCcqhG2VvxQUP4zDehqPATYVZY6yg4Gi2x0cbnmlXfx/P2UTMeVYnkY9ENU+OQDx8J7Mb4DsMfl+40YwnjBt+OBLtXHj3UgcjB7xAW6pjkdmeCKWHt8ekAIzrvgwBE2Z2cThdDUAAA==

LKysrmJubw9/fn8XNMorFzWI0ZcoUuLq6om7durC2tuZum4+kYcOGpeJImLa2Nnr37i3Z3l35+flqV8jv3bsn2aP
0mujy5cuwt7cHAFy/fl3pmhR7ABobGyMtLQ3m5uYwNzfH2bNn0bRpUyQmJqoMDaH/hsMHSsayZctw69Yt1KxZE3X
q1FH5zpbi7rxmzZq9dUCPVFy/fl3RRxh4tVvv9WFILVu2xPjx40VEK5KWLVsiMTER27dvR0JCAuRyOQYMGIDBgwe
rLEDR+1m0aBGmTJmCBQsWoHnz5ip/n+yTVjYV1lPx9etDhw4toTREZYsmLvamp6crFskqV66sEv3+eefY9y4DN3a4cSK
jkUAsbhajiRMnIiwsDE5OTqhWrZokCxOlwZIlS+Dl5YVFixbBxsZG5YGUN8bFo0uXLvj555/x66+/AnhVaHv27Bn
mzp2rNGSBPkxYWJjoCO+lY8eOrOAkKgr29PUaOHAlPT0/s2bMH58+f525D0ihSXVh6G00Z0PP48WPo6PzfLebDhw+
Vrufn5yv14JSazMxM6Onp4euvxYdpdRxcXEBALi5uSndJ8vlcshkMh47LqO4qEdUsuzs7JQ+g2/evibKlRi72WlpZ
ISkqCubk5rK2tERgYiJYtWyIoKAgGBgai45EgLG4Wo61bt2Lv3r1wdXUVHaUK7x7tEspk6C+vnBycoK
1tTWeP3+OwYMH48aNG6hWrpOh4J8uuvvvyqmKI4dOxZVq1ZFREQEevbsibFjxbpwp7ns7e0RehICQ0NDlRv
ON0nxJlPT5ObmAgA8PDxgZmYmOM27adqAnlq1auHy5cto0KCB2utxcXEVotTvGCaqupo1a6J/Ljp
OqaJpC3pERKWRJi7wvm7EiBGIjY1F+/btMXPcMTRs2FB0lFItKzPDPTTzHdnagDRj5mDEN2U9BOA+sFqxDT
80GsxMTH49ttvSy5MKZednY2dO3fiwoULyM/Ph729Pdzd3RVIei/6du3ZVWfd3dSVr2eV1zRhFA3zzv4Ko0Un+z5UcEPMNcpnKaBd4G+g0LcZDuDcAQZ6uGQ/V5bF5ZtJMExAjzngeU9/7ORFLU7wdvJFkXJM7gHk6o/uNQhEhWj5e6pzNmjJFNE6S3pzLIVt1ThrLhxtgndHIIUtrwY5vuPSkX4+JQohCqfzo2zLGrqB9VvF4E9FHMWBanTlz+GKLAhRh+Njxqi6OnMlKL0G80LyXM9LWVWyJu9Qbb7q9/6AqswlmdlzKidNGmxzRHoqcxx7Q22b5GJWlt/L2jHPI1PnqZU3xqCJcqz3tthPMWfrNVb+GQRXJ0TXjYDNaM5LgdY3+G7kD4p+AgO2A+Byi2I0NgpMtJUDj0ol+3IWi1zDmYVZ1/LDC4mRoIvWEOpnC+xOvQBepO5sPlDrJnHrTxiqc6D4xy0LgVfcGz1NZ3q8q0EXLwtf5rh2TPZSTH08P3l2oD7kh/b8Yu5L6r3RcKhU8q5R3DZGZJtuTPI0OPavK6H1Mm5P/BRdBXoaegrHSRWW6bY8GgJNBEYx2HZPDYa2cQLFYv8AZIeW/cPI9xcSZnoyuJ/EyNfzzB+aCNyKt0tDaoRu8UyEWZQUd6wCTw2rFwP6T26aIoOaNHV/+oKaRYf5WIXy7AMwV7kFt86MS2Pd3uKJRl6nKVFEDBjjFkaHXAOlaS7IYLvdfTMgAUFG3Q3FoKOcqiSU5fKl+XIYPYoWiT8yqE4Bd1Ph5BCYG/HXuyJH+F/ijCOj+T8/o+WEfXlJg6ibzj5mNsThAZvNh1xDu34fmpqqaPf3mmdx5kDmeyaXeHfh3LMIWMDI5Xcc3VE5XbKNBjZQ5wQ3WbLP3Md/JGDmEYuwLnVkyFCkgHbpkt9u7dUTx5E/bjfsy+YZGpVhwj72lHA62W0mQFdeIfFMpg+kvmKi1iIlsSyjFHgqyGXhlgkbFv6m2rD6ACzvUxhdtHTZ3UMLUS6gZfmIASJgtXMTCGdqmRmk40yl4u2vrlbkEU8OJ5z2Jrs5I5yMJIR+pcLK0fHjKBVcKZv37hoPvOUNxUeAOZMpuHGyhn/xLdCd4j4K6bb1kESVxdnw5S2X1HNqB0jHdNFDpw0HhY7T2KI+XbIeDEQa8Z8pp2vQUTM/NCqRzXBJeCgUwS8O5yqRd7bRRO4Mb4wT27+XYUzMTq0lhNHTTEp+GGvqqRiSAd6B8mcf9ZB4k1ivc4JYrwmZtVeRSRjgN5/UBZfTB8j6ntPo7HUPQOdEX5/uGSp1yLpA6bX5bsidTpmjVf0bW6ZiLmLQ/AXLLNxXA1QBXwcSQn3+Kr5VVSqxFbwdHZUMM3L8uEqtIGSSEfBnR5bxyhJyTTuqRDpQwUMPdCxz7ilvjfpQ6ajNaa7ICVXjS+IxIJPVRhzGuV+F8c4P2gO+KETsmlYHT/Km2/MS5QFcQ//Pi2oEqFUHMS0ug1y3VF1j++QjnlCL63ExzBbTJlu8AZCgwC6gqf3Xl4fXVaeNnN3vLVq9ZSmqzQYgrdyyMOPXu6RVQVJCAjjCMiNvj3BM8yrhhy06WF6WCnZuZYrjmnwSi/ONXxnPZG7sazEmJCJuCx5WaHRuP8krTwWsMCsZvuTmOS/9hOG2bx/1DdiEXnmYNSUn0hlYXbbbHKcRz+Sv1vfWuhrsyVOdd2umrtcOsRVhw9wt6ySnfx1cZ7GQWCCYN23SjNt/ZvK6iyj1hRKoo0y/1zBt8/t6jlR6c4cyM9vvmBqkCrtRxiEvqo5U5ytGmFxxFqgLLDwC64QKGQT1VnxKwcnv4yP7zwDC7IsIqZVBUr5G1WeS6SXhAzW6YnHqxAI7JQdEUiKF5VUz2I4ovSK+gGD4bvZwKqf5JmcTsZAbJp5rt5y3JFR5cXeoV8RG3nDe2bh1ZSpxuoeoXDz5cHlBHiQhnmAkPeQmsW/nPWpd7IWvAxDEgXLBtBUg1Y9Ejwg8v4ZDQcUfKnHuBlNuVr2Bv8l7kLt3PSEDGvIvXEEvYqsYkbLpC7qpI3t3UUBYRkxC+v1SBXaU4nBTvjyZHQCsbYB5bf8h1P4B8MTWsaPjH8H9ZcNhTDNWECbFbJCYj4Kv6+ZOiqI88VRq7R8y8Tpy1b1f7p1wNTYeB5Pe1Gd6aNavQfKqxkJwmP3dD/cHHDE83fu9J3Kmf/v1HyfiGp6x/R+U8jZq6V+x1U3xHm9Zd0/Lw/uyYnMb2ofR2pv6UbvDKdsS8y0i8qHGAQ6+RXqYcB6azg6ODEkfZ6ynV0hyM5gTzKA7NZWD16h4NXcpgq/HNOZ6RHyuw5uFSZN2E6bf9ptdkeu8KiTTRN2VSkSrKcRzKpqhmOiTHgdLOCD8Gew2jyR+6mAzDQQ/iqmgk+0wqPDhjVHgWI3mm2x5Hc6L+1HKYWg8hdZ8rAGeIIXk16mpDMoB4WMyFZUV

LKysrmJubw9/fn8XNMorFzWI0ZcoUuLq6om7durC2tuZum4+kYcOGpeJImLa2Nnr37i3Z3l35+flqV8jv3bsn2aP

Ij48XHeWdUlNT1U7wLqCjo4OHDx+WYKLSJzY2Fj/++GOh152dneHj41OCiUonFi3pfbxtwBB9HI6OjnBwcBAdg4i
oTHj+/Dl+/fVXnDhxQmOG8/j6+qJ79+4IDw+HtbU1AMDHxwcLFizA4cOHBacjUVjc/EBvayr/+PFj+Pv7Y8OGDYi
NjcW3335bcsGISCO0atUKMTExMDc3Fx3lrT799FNcunQJVlZWaq/HxcXBxMSkhFOVLiwgE4n1Zl/I58+fY+zYsYq
WBQWk0gu5tDIwMODfMRFRCYmLi0OzZs0AAJcvX1a6JtUFvhEjRiAtLQ3Ozs6IiIjArl27sHjxYvzxxx9o06aN6Hg
kCIubH0FoaCj8/Pywb98+mJubo1+/fti4caPoWEQkQd988w28vLxw7949NG/eXOUh2tbWVlAyZd27d8ecOXPQrVs
3fPLJJ0rXsroMXfuXPTo0UNQutKBBWQisapUqaL081dffSUoCRERUcnQ1MniU6ZMQVpaGhwcHJCXl4fjx4+jVat
WomORQOy5WUzu3buHzZs3w8/PD5mZmejfvz/WrVuH2NhYxVZpIqFXpFXpNSr16U1NTYW9vD21tbUyYMAENGjS
ATCZDfHw81qxZg7y8PFy4cAE1a9YUHVVjTZw4ESdPnsS5c+fUFpBbtmwJFy5UlBCIiIiIqKSV9j9r4r4+PD9q
1a4eWLVsqXps0aVJJxSIJYXGzGHTv3h0aMH3N3d4eLiAm1tbZQrV47FTSJ6qzt37rz1upbSOq9+5cwfjq
DsWPHUPDVIZPJ0LVrV6xdIZPJ2JhYVGk98lkMty+ffs
jpyEpYnGzGOjo6GDSpEkYN24c6tWrp3idxhT0tDRUq1YNAHD37l389ttvvyM7OhpubG9q2bSs4nXXqPHj3CzZs
3IZfLUa9ePRgaGoqOVGqwgExEREJSUgIABDhw6Fs7MzgoOD4ezsjGt7AnEFLFUHcrWrTvFpoGdoO9Bz55bTVk
WL18ODw8P6OvrvFp2QcUvRvV36OjoYPr06FIe+fTsOHToEZ2dnbNiwAcCr/ovRE4dE4e/as4KREREREFQa6enp4cq
VK6hTpw6qV6+OsLAw2NjY3dsT9+/dFR1Tr3r17OHjwoNqrZu27YhIyM
DXbp0wcGDB0XHIiKJqF69OkJDAhnzpzpiKgoODAwAgISEBrVu3ll3DkyZNkJSUBLlcDnt7e4SGhoqOSa
wsbFB06ZNMWPGDAwaNAhnzpyBi4sLHj9+LCwXLrl1DkyZNkJSUBLlcDnt7e4SGhoqOSAIUPqa
XPkiDBg2wdOlS3Lt3Dzt37hQdh4kJj09HcbGxgCASpUqQU9PD1WrVrVlVcNzQ0xNOnT0XFIyIiIiKiUq5t27YIDg4
GAPTv3x+TJ0/G/G0a988gn27t2Lu3fvon379Ly8vLb16Sm42qA3aT5WesuBSJSSsP0bnT6OAtSYQQUe8cQeCY9
tpKamokaNGgAAfX19xMXFKSYBpqamwtTUFHo7nz5/D1NQU+fn58PHxQUREBKysrKDB79mx
J9n3X19fHxYsXUbduXgaGiKgKNGdzdGbGwsevXqhnkrHlHENYsepVqQU9PD1WrVrVlVcNzQ0xNOnT0XFIyIiIi
AvHjxQmQ0IiIiIiIq5V4/OaalpYVh2hqurK7y9vXHv3j0MHTpUdCwAQFBgwOYwKVOPU+xkgtv
GAPTv3x+TJ0/G/G0a988gn27t2Lu3fvon379Ly8vLb16Sm42qA3aT5WesuBSJSSsP0bnT6OAtSYQQUe8cQeCY9
tpKamokaNGgAAfX19xMXFKSYBpqamwtTUFHo7nz5/D1NQU+fn58PHxQUREBKysrKDB79mx
J9n3X19fHxYsXUbduXRgaGiIgKNGzdGbGwsevXqhaSkJNERSQAd0QGIdMqFCgCA58+fY+zYsdDT
AvHjxQmQ0IiIiIiIq5V4/OaalpYVh2hqurK7y9vXHv3j0MHTpUdCwAQFBgwOYwKVOPU+xkgtv
M6eevvvpK5T1Dhw4tqThERERERFQG5efn4+n/4+bNm3jw4AEcHy8vLdPEKCJE
s27cPrVu3Fh2PBOGxdCIiIiIiKiMubs2bMYPHgw7ty+zEZTJItsm7fvo1nz57Voiuva+
vL8zNzUVHJAFY3CQiIiIiIiKmOaNWuG+vXrY/Ly3NWKURQ0IYICEGWHHjh2wtrbG/
jYWVlJTpKkd29excCxaeRFRWHHjh2wtrbG119/ZTNm6J0BrJfsceMtjRRmLCSbBNnrbN7y2FLb8
jLCwMAJCSkoLOnTsjJKNioKs2bNgre3t+h4RCQgIeFBjRcHVTMkeITPAWKBeN1wU3eQGEIhWrIoKgm
vX0bLli0BAIGBgbCxsUFkZCHrIUtWrhkEunwSkYuOnmiHg5XDzrz8JUHDeZQzYHRfd4CCrfP/JgkoNgQlXYGm
TgwoVKgAATpw4BAw4ATc3NwBAw4YNcf/+fdExisx+/XChCJdw1fFsddvxJkYSGbNiQ69R7aRm0FyajxLN6fP7J
zzz+oVq2a4HQkCgcKERERERERERGRGR5J08eRkJwZPEAZs2ahSZNn
iojLq6tWrSE5OxsuLUL5VeLzjyTV5eXl48BSVBV1cXRkZGQrtSOLAxA6VNFa8lJSXBRKVZjjAlLwb
w6dIlRa9N4FXfTQl7Lw8bhIRERSFGwkmG6Yg8dcdBKJICEgSGERo6ymCEvF4y9e29Oy6TBsCeXQNQ3mF
hMDQ0hJ2dnaL4pzJMuPTqLpDNNRA8dqS6fKPkBvIqD3XMCEAHxNfHUzhVHRbvU9Pv2ZCzYJXXYRlBGRJER0
RANCrVy9cvXoVjo6N27t+g4JEEsbhIRERERERERGREV0Sf8a8rTZ0KbBaA5ibm+PatWuZ0sCJq4IiMIERg
nQktLC3Z2dhg5ciTc3d1RpUoVw9bFIQrREByAiIiIiIiiopLVpExMIXFAQBatWqFpUuXIjIyEt7e3CBS
cTll6c9kzKY8YCtwl/CERjRJrj5Q4Jw2Ykja2OtE7kQCfYixfDjbVkR6jRKIW6Q2h9a6wb5XrwO0sX0i
+BTN3mnZaJJUlb1vODGRmZoR4jl8mQNwy6sWeFrRFgRHoL8jsunWyMYxC9Z29eQ8RlN2cQJYrGvCQk
Zj0KDJeGYfKzQKcB9wjEEKHjqFeI4kzCRpGXbIW6qdp+5wdJGJHuBT6U1JZI8U1KpJvz6dQNqwiGQHcv3B1V0t
FbKs8MSRGmSJbfI+Cn+9DtuHf5bJUmBcZnHGmGxKtjFc8jZmPezEhGXBJxNoGBbETWjN+FhfaFK4iDpFHJc+Odx
KJqaKIlbfGbdtE5b4FPIT6aaiGk3rbThBBEaK+jcTnVx6uxf27R5xVGNFFUBdmHOFwOVRXG7rnNMUkWzqqKyjrKjZ
rlvMcLp0nVuGPZjNeMhCCDRRaLrHxCNa3YnEYtBFxNIb8gFqXzx/qMbStE9ymbr3lcAjNFLjY9mT7Mf1rH3NxaX
d1GBOkAQe51dDhElhbRTlF5WYjSjMDXIJLUjCWMhj28fR3BsEdAWjZDjk7O4D2GKa5Fj8hDqMBA+Cz+NCiHgVhZ
+YGDW/ZKWjT0n+fb/fc3i4RnY8oWF8Xw7pRDOPpAnqUiA1kWVHJHvHwqPuEUjl0rHwG6uO+RjKAlmGbHjW1l6u
7fZYmb6Dd7d3nVHqXJMCKqKPXfLPd1I3QPFnPNAGC6oKzgC0N/KmNrHi7SoO5jNFXn1b3mmFnnYkCfa6VRKTqAN
TOtRNkgSbNP7hX0JXYd/PJPbEHe1llNJDHoRmm3G9mNpeVHXU3AokUGWK29OgtO1Xg6MjULeIUafx9gRTVmMEd
ToB5AZR/oWG8x5FVcB6Yw0Yk8sMtLAG6LFQC3Ysn2WzPWkvSz7F6k4uhXqBbMlp2LBa4YV2fdOzOV0dQS8pMUu
UCGhDlPgp0qPGYwg+ZT/S4c4TzKN7pQqd+iq4xknCKNF8IOAjDJ3TEPJlNwJuzC3n+iJ5vhrXd5ZVx5xjSwpBvR
W01vbKfb5UWoMRQb5nNLRUzfc3R3fV5/VKFv/Vv8Ri+kfJPBuKM0+8aK9+gm73CpcnB+kPpkvVy5W9+ZZmIrk3tM
hSyHnxS9YT1oa5qOxHJ21wzA9O+9X83MwfRURVQLBYdKKsdGsomjqkmDwN7VsIaWbbIQRVrJjDHNK78BKQZGWMk
LC+jfXnvWQd7FGn+XF4jEl6bNG2m9P6EqUWydf8J9Hv8sqxX2k8i3aWfcHNlmc/pP6sLhKgD6PSHFGCaK6t0K/UZ0
4g0bBTYMMXgd/m9/7uvVmvHj9i+b7TpJZ7BXrXhWiW3D9+cvBXKZXXJw9TVOCZlV4rcWpCNB8SMyiGFoTt2p0B4P
FR2Jk0hjy6cjbWGGhSHcAA3qwy2KQPEQnwTKJRyAvHkTvG7F1L2WtBvWqmNl5eDHpdVwJTrd8JJK0Mwm7vdEGd
+SWuvZIhNMfrmWT3WL7n1OUIjv+YJKbCOXLNzyyR8vz9rA7J5jZ/n9Oxd3cDnu5O5VVfKyfqJ3XpjvKnLIf4EmM
XN/9KOFuaefujRedSOmCRvYXzpZs9VObZCmQqdgmP8FtxWtFhq1HthSQpc8Hn+bvDJCe+vnxmwFbmMHHJJwOu
DXeoMAlWRaU8BO5bbqAg4QLXyQXSwgV2Z8M71hmDKQxoPRUcPvaC4gnMf9VtWd93GUV4dnPhH7B3dE2vKY5M2jT
+ccmbfncHBPmlvGwMJP+MDZhm8wYPhbZzNHt3HxxXFhqbhPbmPNHHOaNsk9QaKwRCnaMQbnmXbK5wa89VTtF
LhY3sknV2kYxbm56b0PlzVcXYFGyLs3b4WFXqZd0lSXsdOuD2HPmrSo2Rps9xrHHhAQzcKKwRT3hE6fjmOU3+YAY
xAZR4VSeAdF6Ltqg3B2Rq2y2N4y7nGv6Kh6sq5DkhKu0MTmnnjAaMvvFRyHL5S4IR3t1UhhZrW/p7dRjoS3Ip
oykP2GGbcWKOkbQwVkiDGLSeXj7j5AdZmTTZ3b6BXf8AQY6FKRQ5kRIsx6ZKzm8ngVfvFSfUcnYDpyLN7lqjjM
fVPR75bnBcLAABIyTkbqQKrmktAZc8F4ZGE7oFHmLKm26RXNBqxhBnhkTyRdnCeIUk1aYVBDqaKEbt0Rv0Eq/Md2
r3tzbJ5JDAgIPbmjmPqJWzk4AJ6FJdX5ZaSyuuXm3ezS8VQqiPnKLeHAVKrYM0sVdh2QhjT6WqzVRHAakknmUaEjF
38nZkpE6vh8AOsnqdUbThG8tXe26v5A2OJ8zeaTvC45JMnrB3m9dnPrOx31eq8Q5xFAUXdfYqPrOK65M3vBTtJ7KU3
fOvxN8N2OsK78g6CxuzfSjtX7DMOXeMfoyD9i3HP+6fBLmdWtZC6wJc2x2BKUdx2ZLYxL0NMYX3CAaPLjXs6vdvn
cnjbTkoqbw+Hc2TWMVzRh1LuR4K6aZ3tXDVFVDWK61vwWnv3P5XthJGsVQaUzr7NfXTHqfr0tKE2UzX0hM2Sku1
xqsE/bK1rU3pRoKbC0HXe6Ux7jkCgHtUOn6NmmmaC3R0I7yDaFAXBwMNH7XPRiPhNRgCttjFkVW3FsPF1vJTXbFrp
U5Zwp2fDUKV6X5Wdw66XE3XuLQcN6OX7X9xpQXj4lG3ooPNPMNbRd5wjK3KPb+NvLV8OiJhdrnFp9kbGGqXxScdY
e34MfLmR9Fk0qYRrd+dCnSvuxmVKo53nyZgdQJOd8mWy9bQKPL2b6wJEHSBVEEmKJFHrRs7DvK9jaupf3fF3kHwx
RJqT3R2E7xeaZa7c7AR7NxiYzc2HWMJjD5zNemzjTbn3VgG3Ja9wwR6pj6n6Q8kuzXYxw1zeZtF8n8YnR7QHm3v0
VfXNLu9Am1ALd5J2Rl6z7uQlE4M6KL4K9E7EjhE9MFdkNnXVdHqnJh9HdN8xmrD2cqWEpK7bBbYbcN93pjLb9+U
/6aHN0JmEzcf/Ldl85Nt3nXAP5fJafl7zsQTr3Ls2ZU+SbyLHdDAf5c8gzFLHqaEEuO2jd1pQ4Bbmrc+aEJLnQ6Vk
HpYAmS+Ot2QdyQ69Q3zRv9e6ENdrdBuM6S8ZCrrzmVmx7e6Odpl4c+VHZtg5vfj2wTtL8dxGhe0qcW+fEbM2Qn
OoUwVWrz1j1z5rUybNnHD6xOj8NvSqYAAo6NR3p4c9fJbmrYcmaCNvq9GbSNsFxdcxCmqk4mtZTb6CnaF1dxbxs
OtT8/zw8dbQWHrtvAb1WiQdYzfXGVwqhAgwmaB2zvb3Ee9TvXHOdbkuF4RnGbGaRLIqLWy8dnIlGUW8RM0w5hOaK
b8cPWrqEu3qfRxZAFUP8OX8BnfUOTfMLXDj1BzTKULXJ69XLbyrdwqsNvhPXrRvbOb6hGcA+lKZ3zT8BgebtLs4z
6HTXWwNYeU6A3bH1ja3+O81Q6Nh2ZJsrDlGoHfNXtvJnZLtOa2HEdrDCbpMGmTlKEDN7kZ2DFwoq+wJHTPxDL9
yWLH7wNQPNvF6bYsFtbJxQ+ZDn3aUBAR+eNdwzMEz1jh7lqHEmQMQHJ3x0h0JeA8sXtqNDTzjeRWm0U3HS+4+Wp
u4QvT69XPZsFXZnQVtzEhtJ9OVWTv2r0ecRVThZOVPdb1Unt36ByHXq7UXT81gGFJbEvpevT3n64VUfmB5RYHtWB
Kl7TSwDyNTCqzMBUT8y6B1Fm2zxYWVS0jUvHzLSRZlWGXDTzR8j7b5v12JqH8XeQ4e7cS1hQpT4HWhdR5CSSgD+V
Nl8gDK3WuAVCvvGeaOP9bTL1FHpBUHQjR5YwD0pHq3yl8qRtGpE0XkMYlH5mTdMRtmxzv6ejvXW0n1OMLqH2hjdQ
S9ZtV/VJeu9y1VS3o5jr9/ZxmYjOJM8U5Yne5wWhuFdy8nUVp9F0l5UDgvF8YZ1ZxSnWv8DIcjYYHc4V+wOVW+MX
LHuBBQtl+nlHH7jgZr7uZnTfOzrnR4HT1+Mmxy3N7TjUKV4Q3+LnqSVm1LKq36JxL5XxPV3tbNcVX+Ojp/C1aQ8jvV
/FpyHqXhrkfTQlzD5Hxn63pQeEUy2Zv6TP8d3ibMv7b7WtlZNd8p8oX8JRbQVT5RvZUUrNXDsTtlHbQt7nmW+FRf7
Ij48XHeWdUlNT1U7wLqCjo4OHDx+WYKLSJzY2Fj/++GOh152dneHj41OCiUonFi3pfbxtwBB9HI6OjnBwcBAdg4i
oTHj+/Dl+/fVXnDhxQmOG8/j6+qJ79+4IDw+HtbU1AMDHxwcLFizA4cOHBacjUVjc/EBvayr/+PFj+Pv7Y8OGDYi
NjcW3335bcsGISCO0atUKMTExMDc3Fx3lrT799FNcunQJVlZWaq/HxcXBxMSkhFOVLiwgE4n1Zl/I58+fY+zYsYq
WBQWk0gu5tDIwMODfMRFRCYmLi0OzZs0AAJcvX1a6JtUFvhEjRiAtLQ3Ozs6IiIjArl27sHjxYvzxxx9o06aN6Hg
kCIubH0FoaCj8/Pywb98+mJubo1+/fti4caPoWEQkQd988w28vLxw7949NG/eXOUh2tbWVlAyZd27d8ecOXPQrVs
3fPLJJ0rXsroMXfuXPTo0UNQutKBBWQisapUqaL081dffSUoCRERUcnQ1MniU6ZMQVpaGhwcHJCXl4fjx4+jVat
WomORQOy5WUzu3buHzZs3w8/PD5mZmejfvz/WrVuH2NhYxVZpIqFXpFXpNSr16U1NTYW9vD21tbUyYMAENGjS
ATCZDfHw81qxZg7y8PFy4cAE1a9YUHVVjTZw4ESdPnsS5c+fUFpBbtmwJFy5UlBCIiIiIqKSV9j9r4r4+PD9q
1a4eWLVsqXps0aVJJxSIJYXGzGHTv3h0aMH3N3d4eLiAm1tbZQrV47FTSJ6qzt37rz1upbSOq9+5cwfjq
DsWPHUPDVIZPJ0LVrV6xdIZPJ2JhYVGk98lkMty+ffs
jpyEpYnGzGOjo6GDSpEkYN24c6tWrp3idxhT0tDRUq1YNAHD37l389ttvyM7OhpubG9q2bSs4nXXqPHj3CzZs
3IZfLUa9ePRgaGoqOVGqwgExEREJSUgIABDhw6Fs7MzgoOD4ezsjGt7AnEFLFUHcrWrTvFpoGdoO9Bz55bTVk
WL18ODw8P6OvrvFp2QcUvRvV36OjoYPr06FIe+fTsOHToEZ2dnbNiwAcCr/ovRE4dE4e/as4KREREREFQa6enp4cq
VK6hTpw6qV6+OsLAw2NjY3dsT9+/dFR1Tr3r17OHjwoNqrZu27YhIyM

REVEZo6enh9jYWFhZWYmOQvRBtEQHICIiIiIiIiKiktWqVSvcvHlTdAyiD8aem0RERERERERERZUBcXJzizxMnToS
XlxdSUlJgY2ODcuXKKb3X1ta2pOMR/Sc8lk5EREREREREVAZoaWlBJpOpDBAqUHCNA4VIk3DnJhERERERERFRGZC
YmCg6AlGx485NIiIiIiIiIqIywsPDAytWrIC+vr7oKETFgsVNIiIiIiIiIqIyQltbG/fv34eRkZHoKETFgtPSiYi
IiIiIiIjKCO5xo9KGxU0iIiIiIiIiiojJEJpOJjkBUbHgsnYiIiIiIiIiojNDS0kKVKlXeWeBMT08voUREH4bT0om
IiIiIiIiIypD58+ejSpUqomMQFQvu3CQiIiIiIiKiO0tLSQkpLCgUJUarDnJhERERERERFRGcF+m1TasLhJRER
ERERERFRG8AAvlTY8lk5EREREREREREQaiTs3iYiIiIiIiISCOxuElERERERERERERQaicVNIiIiIiIiIi0kg
sbhIRERGVcTKZDL///rvoGERERERE743FTSIiIqJSLiUlBRMnToSqVKgAMzMz9OzZEyEhIQCA+/fvo1u3bgC
ApKQkyGQZXLx4UWBiIiIiqKi0REdgIiIiIg+nqSkJDg6OsLAwABALly6Fra0tcnJycOzYMYYMYwfPx4JCQkwNjYWHZO
IiIiI6D+RyeVyuegoegQREREREPRxdO/eHXXxcbh27Rr09PSUrmVkZCMDDAwAABl7ye4EI0ZMdFSrt5mndgXpIUcqQ9G5NTZ
nTrh7965SIdTLywvnzp3DqVVOnSuR/CxEREHRm3gsnYiIiKiUSk9Px9GjRzF+/Hi+VwViYAGBgYqLwWFRUFADhx4gT
u37+Pffv2oV27drC0tMS2bdsU78vNzcX27dsxYsSIj5afiIiIiOhdWNwkIkqVu3rwJuVyOhg0bFvm/U6U6N3E+/Xm3i6
AtWrVYGxsjKpVqwIAe4ciU2bNined/jwYWRlZaF////7FG5qIiIiI6D2wuElERerUShV0H3rzqPl/MXz4cNy8eRN
nz54FAPj5+aF///5qd4R4QSEREREZUUFjeJiIiISql69epBJpPhMhPj7+g/9vGRkZoWfWfWfPnti0aRMePHiAI0eOxh
SEhERERH9dyxuEhERSVatWuWdXbFmzRpkxmzcjVnyiaOmnaMSTnbP0ZmagXM/IyFB5rXz58gCAvLw0bH
YMxMRERERvQ8WN4mIiIiIiIgfhKsbvr17+cePGDcTHx2PlyWPPVN5vzyGSEihUr4ujRo0hNTcxjx48
V17p27YoqVapg4cKFHCRERERERJLA4iYRERGRFRKWZhYELFy7y7AyckJXJ5eaNKkCYk6b6+d0YKQBD88sKu/X0dHBU
rsX79epiamqJXr16hwg+fDjy8vIwcXhZRaSmqJRTVRr4VFQg6U1lkxe0gmeiIiiIOgdRo8ejdTUVBw8fB0FCIiI
i6IgOQEREREREREETS9/jxY5w7cDhERERERERERABY3CQWDhmefz7vvVXZGSEihUr4ujRo0hNTcxjx48
ORERERERERERERGooDhYiIiIiIiIiIiIiiIiIEgjsbhJREREREREREREERGonFTSIiIiIiIiIiIiItJILG4SER
xk4iIiIiIiIiIiDQSi5tEREREREREREREREREESkkVjcJCIiIiIiIiIiIio3E4iYRERERERERERERERERFpJBY3iYiIi
9P6szdHUkfBlUAAAAAElFTkSuQmCC",
"text/plain": [
"<Figure size 1500x800 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plt.figure(figsize=(15,8))\n",
"sns.barplot(x='City', y='AQI', data=city_median_AQI_per_year_test,hue='year').set(title
='City vs Median AQI per year')\n",
"plt.xticks(rotation=90)\n",
"plt.legend(loc=(1.01, 1))\n",
"plt.show()"
]
},
{
"cell_type": "code",
"execution_count": 43,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"   .dataframe tbody tr th:only-of-type {\n",
"   vertical-align: middle;\n",
"   }\n",
"\n",
"   .dataframe tbody tr th {\n",
"   vertical-align: top;\n",

```
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>year</th>\n",
"      <th>AQI_predicted</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2019</td>\n",
"      <td>497.356934</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Ahmedabad</td>\n",
"      <td>2020</td>\n",
"      <td>181.523926</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Aizawl</td>\n",
"      <td>2020</td>\n",
"      <td>56.157894</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>Amaravati</td>\n",
"      <td>2019</td>\n",
"      <td>74.475441</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Amaravati</td>\n",
"      <td>2020</td>\n",
"      <td>68.160255</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>5</th>\n",
"      <td>Amritsar</td>\n",
"      <td>2019</td>\n",
"      <td>96.089272</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>6</th>\n",
```

```
"      <td>Amritsar</td>\n",
"      <td>2020</td>\n",
"      <td>90.253029</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>7</th>\n",
"      <td>Bengaluru</td>\n",
"      <td>2019</td>\n",
"      <td>88.249664</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>8</th>\n",
"      <td>Bengaluru</td>\n",
"      <td>2020</td>\n",
"      <td>77.836243</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>9</th>\n",
"      <td>Bhopal</td>\n",
"      <td>2019</td>\n",
"      <td>158.006180</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>10</th>\n",
"      <td>Bhopal</td>\n",
"      <td>2020</td>\n",
"      <td>110.654266</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>11</th>\n",
"      <td>Brajrajnagar</td>\n",
"      <td>2019</td>\n",
"      <td>127.240646</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>12</th>\n",
"      <td>Brajrajnagar</td>\n",
"      <td>2020</td>\n",
"      <td>158.138916</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>13</th>\n",
"      <td>Chandigarh</td>\n",
"      <td>2019</td>\n",
"      <td>113.270340</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>14</th>\n",
"      <td>Chandigarh</td>\n",
"      <td>2020</td>\n",
"      <td>79.965584</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>15</th>\n",
```

```
"      <td>Chennai</td>\n",
"      <td>2019</td>\n",
"      <td>84.210533</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>16</th>\n",
"      <td>Chennai</td>\n",
"      <td>2020</td>\n",
"      <td>88.648903</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>17</th>\n",
"      <td>Coimbatore</td>\n",
"      <td>2019</td>\n",
"      <td>88.451118</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>18</th>\n",
"      <td>Coimbatore</td>\n",
"      <td>2020</td>\n",
"      <td>98.433853</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>19</th>\n",
"      <td>Delhi</td>\n",
"      <td>2019</td>\n",
"      <td>220.540451</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>20</th>\n",
"      <td>Delhi</td>\n",
"      <td>2020</td>\n",
"      <td>152.999207</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>21</th>\n",
"      <td>Ernakulam</td>\n",
"      <td>2020</td>\n",
"      <td>101.220333</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>22</th>\n",
"      <td>Gurugram</td>\n",
"      <td>2019</td>\n",
"      <td>167.739212</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>23</th>\n",
"      <td>Gurugram</td>\n",
"      <td>2020</td>\n",
"      <td>136.080643</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>24</th>\n",
```

```
"      <td>Guwahati</td>\n",
"      <td>2019</td>\n",
"      <td>100.264885</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>25</th>\n",
"      <td>Guwahati</td>\n",
"      <td>2020</td>\n",
"      <td>139.772034</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>26</th>\n",
"      <td>Hyderabad</td>\n",
"      <td>2019</td>\n",
"      <td>93.330879</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>27</th>\n",
"      <td>Hyderabad</td>\n",
"      <td>2020</td>\n",
"      <td>81.613045</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>28</th>\n",
"      <td>Jaipur</td>\n",
"      <td>2019</td>\n",
"      <td>115.021461</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29</th>\n",
"      <td>Jaipur</td>\n",
"      <td>2020</td>\n",
"      <td>109.830193</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>30</th>\n",
"      <td>Jorapokhar</td>\n",
"      <td>2019</td>\n",
"      <td>115.822853</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>31</th>\n",
"      <td>Jorapokhar</td>\n",
"      <td>2020</td>\n",
"      <td>136.280579</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>32</th>\n",
"      <td>Kochi</td>\n",
"      <td>2020</td>\n",
"      <td>86.128380</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>33</th>\n",
```

```
"      <td>Kolkata</td>\n",
"      <td>2019</td>\n",
"      <td>98.974945</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>34</th>\n",
"      <td>Kolkata</td>\n",
"      <td>2020</td>\n",
"      <td>93.204521</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>35</th>\n",
"      <td>Lucknow</td>\n",
"      <td>2019</td>\n",
"      <td>190.335403</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>36</th>\n",
"      <td>Lucknow</td>\n",
"      <td>2020</td>\n",
"      <td>145.659119</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>37</th>\n",
"      <td>Mumbai</td>\n",
"      <td>2019</td>\n",
"      <td>92.529053</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>38</th>\n",
"      <td>Mumbai</td>\n",
"      <td>2020</td>\n",
"      <td>88.552422</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>39</th>\n",
"      <td>Patna</td>\n",
"      <td>2019</td>\n",
"      <td>175.100677</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>40</th>\n",
"      <td>Patna</td>\n",
"      <td>2020</td>\n",
"      <td>130.012024</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>41</th>\n",
"      <td>Shillong</td>\n",
"      <td>2019</td>\n",
"      <td>61.801003</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>42</th>\n",
```

```
"      <td>Shillong</td>\n",
"      <td>2020</td>\n",
"      <td>72.794922</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>43</th>\n",
"      <td>Talcher</td>\n",
"      <td>2019</td>\n",
"      <td>114.360558</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>44</th>\n",
"      <td>Talcher</td>\n",
"      <td>2020</td>\n",
"      <td>116.773666</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>45</th>\n",
"      <td>Thiruvananthapuram</td>\n",
"      <td>2019</td>\n",
"      <td>73.767097</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>46</th>\n",
"      <td>Thiruvananthapuram</td>\n",
"      <td>2020</td>\n",
"      <td>69.124825</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>47</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>2019</td>\n",
"      <td>98.667175</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>48</th>\n",
"      <td>Visakhapatnam</td>\n",
"      <td>2020</td>\n",
"      <td>83.914513</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"</div>"
],
"text/plain": [
"                  City  year  AQI_predicted\n",
"0            Ahmedabad  2019     497.356934\n",
"1            Ahmedabad  2020     181.523926\n",
"2               Aizawl  2020      56.157894\n",
"3            Amaravati  2019      74.475441\n",
"4            Amaravati  2020      68.160255\n",
"5             Amritsar  2019      96.089272\n",
"6             Amritsar  2020      90.253029\n",
"7            Bengaluru  2019      88.249664\n",
```

```
"8            Bengaluru  2020      77.836243\n",
"9               Bhopal  2019     158.006180\n",
"10              Bhopal  2020     110.654266\n",
"11        Brajrajnagar  2019     127.240646\n",
"12        Brajrajnagar  2020     158.138916\n",
"13          Chandigarh  2019     113.270340\n",
"14          Chandigarh  2020      79.965584\n",
"15             Chennai  2019      84.210533\n",
"16             Chennai  2020      88.648903\n",
"17          Coimbatore  2019      88.451118\n",
"18          Coimbatore  2020      98.433853\n",
"19               Delhi  2019     220.540451\n",
"20               Delhi  2020     152.999207\n",
"21           Ernakulam  2020     101.220333\n",
"22            Gurugram  2019     167.739212\n",
"23            Gurugram  2020     136.080643\n",
"24            Guwahati  2019     100.264885\n",
"25            Guwahati  2020     139.772034\n",
"26           Hyderabad  2019      93.330879\n",
"27           Hyderabad  2020      81.613045\n",
"28              Jaipur  2019     115.021461\n",
"29              Jaipur  2020     109.830193\n",
"30          Jorapokhar  2019     115.822853\n",
"31          Jorapokhar  2020     136.280579\n",
"32               Kochi  2020      86.128380\n",
"33             Kolkata  2019      98.974945\n",
"34             Kolkata  2020      93.204521\n",
"35             Lucknow  2019     190.335403\n",
"36             Lucknow  2020     145.659119\n",
"37              Mumbai  2019      92.529053\n",
"38              Mumbai  2020      88.552422\n",
"39               Patna  2019     175.100677\n",
"40               Patna  2020     130.012024\n",
"41            Shillong  2019      61.801003\n",
"42            Shillong  2020      72.794922\n",
"43             Talcher  2019     114.360558\n",
"44             Talcher  2020     116.773666\n",
"45  Thiruvananthapuram  2019      73.767097\n",
"46  Thiruvananthapuram  2020      69.124825\n",
"47       Visakhapatnam  2019      98.667175\n",
"48       Visakhapatnam  2020      83.914513"
]
},
"execution_count": 43,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"city_median_AQI_per_year_pred                                                    =
X_test[['City','AQI_predicted','year']].groupby(['City','year']).median().reset_index()\
n",
"\n",
"city_median_AQI_per_year_pred['City']=le.inverse_transform(city_median_AQI_per_year_pre
```

```
d['City'])\n",
"city_median_AQI_per_year_pred"
]
},
{
"cell_type": "code",
"execution_count": 44,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABTcAAANaCAYAAABcDngdAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjYuMiwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy8o6BhiAAAACXBIWXMAAA9hAAAPYQGoP6dpAADTRklEQVR
4nOzdebxd0/0//teOZFEgiRCIkEMEUPEFIoEMZdWlYoSU81DKij1MRNEG0qotoZQUympoZUKjbSmitQQ1FhEKlc
MkVkSyfn94ZfzdWW6qZvcu3k+H4/zeDh7r7POe28nZ3jdtdauKJVKpQAAAAAAFMD/QrJAAAAAAAMD/QrgEHGG6l
AAAAAA8I0ybyHG6GlXr15WXXXVNGzYsFojDlm4UqmU2bNn/MPP8zcuXPTqTsTmtdVWS8vFbHITkzuUpoQUAAAA
zd+7c2cbxd0/0//teVOZFEgiRCIkEMEUPEFIoEMZdWlYoSU81DKij1MRNEG0qotoZQUympoZUKjbSmitQQ1FhEKlc
MkVkSyfn94ZfzdWW6qZvcu3k+H4/zeDh7r7POe28nZ3jdtdauKJVKpQAAAAAFMwKtV0AAAAAAAKCuAHBGG6lXr15WXXXVNGzYsFojDlm4UqmU2bNn/MPP8zcuXPTqTsTmtdVWS8vFbHITkzuUpoQUAAAA
```

apGnTptl2223z6KOPVmkzf9r0v/71r+y3335p1arVIoO3d955J/Xr119o7X//+99TUVGRu+++u7ztjTfeSN++fdO
mTZs0atQoG2ywQa655poqj/vss88yYMCAbLrppmnZsmVat26dnj175r777lvgOSoqKnLCCSfkuuuywYbbJBGjRo
tNGCer1OnTtlrr70ybNiwbLzxxmncuHHWWmutXHXVVVXazX9t/f73v8+AAQOy+uqrp1GjRnnzzTerfR6T5M9//nM
23XTTNGrUKJ07d84vfvGLRdb11Wnpn376aQYMGJC11lorjRo1Sps2bbLHHnvk1VdfXex7yNKc6yR59dVXs9tuu6V
p06ZZZZVVcswxx5R/EAEAtWf48OE59NBDs+GGG2aTTbJTTfdlHHjxmXMmDFJvvij7pVXXpmzzjor++67b7p165a
bb745M2bMyO23354k6datW+65555897vfzdprr50dd9wxF198cR544IHyH23//e9/Z/jw4bn++uvTs2fP9OzZM7/
73e/y4IMP5rXXXvtaxyDcBIA6au7cufnb3/6WHj16pEOHDkv9+NVWWy3Dhw9PkhxxxBF56qmn8tRTT+Xxss8/Odtt
tl+7duy80hBgyZEi22GKLbLHFFgvtd5NNNkn37t1z0003LbBv6NCh5XAkSQ4++OD86U9/yjnnnJOHH344419/fXb
eeed8/PHH1TqGFi1aZL/99suNN95Y3nbHHXdkhRVWyAEHHLBA+3nz5mWfffbJpZdemr59++bPf/5zLr300owYMSK
9evWqMtruJz/5SX7xi1/kkEMOyX333cf/OAH2XfffTNp0qQl1vXOO+9kvfXWy5VXXpm//vWvqWyyyzJhwoRsscU
W+eijjxZof+SRR6ZBgwa5/fbbM2jQoDz22GP58Y9/Dz88MPZZ599suqqq6Zfv35588038/e//73cZu7
cuRk5cmQ233zzrLHGGgvtp0OHDunRo0ceeeSRzJs3r1rP/VVHHHFEVlhhhdx+++258sor88wzz6RXr1759x
uvfXW7LLLLmnRokVuvvnm3HXXXQvn3HXXXhQZz++67b9ZZZ53cfffdue666xb6vJ06dcree++d6667LnPnzq2
yb8iQIWnfvn2+//3vJ0leeeWVbLHFFnnppZfyx4Mg8++GD23HPPnHTSSeXp1Ukya9asfPLJJzn11FPzpz/9KXf
ccUe+853vZN99980tt9yyQA1/+tOf8utf/3cvLr300k6cbnuuuuywAMPpE2bNtU+j48++mj22WefDm6NGjM3r06MW
OSkuSww47LCeddFJef/31rLvuukm+WEPnvvvuywknnFD+EvPEE0/kyCOPzE9+8pPyY/fZZ5+lOo7DDz88vXv3zss
vv5wNN9wwN954Y374wx8udL3Nu+66K8OHD88999yTfffdt99kbbJGhQ4fm2GOPzauvvpqbb745P/3pTzN
o0KAkSZ8+fdK2bdscdNBBS3e/32y3777Ve+P3fu3Oy1115p27Ztbr/99px00klV2u+2225VRu198sknOf3001N
ZWZl27dot9rluuummzJs3rzyt5/DDD8/FF1+cG264ITvssEOZWZl27dot9rluuummzJs3rzyt5/DDD8/FF1+cG26
BB/K9730vyRdToIYNG5azzz67/Jo75ZRT0rx58zz++OPlqQT69OmTWbNm5dJLL81UVq1apWXLllUCwLlz52a
nnXbKpEmTcNtlkkyTJ7rvvnokTJ+bCCy/Mcccdl3Nm5bJLL81JJ52UVq1apWXLlUCwLlz52aDmpsAUEd99NFHmTFjRjp37vw/Pb5Ro0bp0aNHhkmSNNdbI1ltvXWX/SSedlMTTJ6NGjM3r06MWOSkuSww47LCeddFef/31rLvuukm+WEPnvvvuywknnFD+EvPEE0/kyCOPzE9+8pPyY/fZZ5+lOo7DDz88vXv3zss
yb8iQIWnfvn2+//3vJ0leeeWVbLHFFnnppZfyx4Mg8++GD23HPPnHTSSeXp1Ukya9asfPLJJzn11FPzpz/9KXf
cccUe+853vZN99980tt9yyQA1/+tOf8utf/3cvLr300k6cbnuuuuywAMPpE2bNtU+j48++mj22WefDm6NGjM3r06MW
OSkuSww47LCeddFef/31rLvuukm+WEPnvvvuywknnFD+EvPEE0/kyCOPzE9+8pPyY/fZZ5+lOo7DDz88vXv3zss
vv5wNN9wwN954Y374wx8udL3Nu+66K8OHD88999yTfffdt99kbbJGhQ4fm2GOPzauvvpqbb745P/3pTzN
o0KAkSZ8+fdK2bdscdNBBS3e/32y3777Ve+P3fu3Oy1115p27Ztbr/99px00klV2u+2225VRu198sknOf3001N
ZWZl27dot9rluuummzJs3rzyt5/DDD8/FF1+cG26BB/K9730vyRdToIYNG5azzz67/Jo75ZRT0rx58zz++OPlqQT69OmTWbNm5dJLL81UVq1apWXLllUCwLlz52a
nnXbKpEmTcNtlkkyTJ7rvvnokTJ+bCCy/Mcccdl3Nm5bJLL81JJ52UVq1apWXLlUCwLlz52aDmpsAUEd99NFHmTFjRjp37vw/Pb5Ro0bp0aNHhkmSNNdbI1ltvXWX/SSedlMTTJ6NGjM3r06MWOSkuSww47LCeddFef/31rLvuukm+WEPnvvvuywknnFD+EvPEE0/kyCOPzE9+8pPyY/fZZ5+lOo7DDz88vXv3zss
vv5wNN9wwN954Y374wx8udL3Nu+66K8OHD88999yTfffdt99kbbJGhQ4fm2GOPzauvvpqbb745P/3pTzN
o0KAkSZ8+fdK2bdscdNBBS3e/32y3777Ve+P3fu3Oy1115p27Ztbr/99px00klV2u+2225VRu198sknOf3001N
ZWZl27dot9rluuummzJs3rzyt5/DDD8/FF1+cG26BB/K9730vyRdToIYNG5azzz67/Jo75ZRT0rx58zz++OPlqQT69OmTWbNm5dJLL81UVq1apWXLllUCwLlz52a
nnXbKpEmTcNtlkkyTJ7rvvnokTJ+bCCy/Mcccdl3Nm5bJLL81JJ52UVq1apWXLlUCwLlz52aDmpsAUEd99NFHmTFjRjp37vw/Pb5Ro0bp0aNHhkmSNNdbI1ltvXWX/SSedlMTTJ6NGjM3r06MWOSkuSww47LCeddFef/31rLvuukm+WEPnvvvuywknnFD+EvPEE0/kyCOPzE9+8pPyY/fZZ5+lOo7DDz88vXv3zss

2226rMoX6ySefzLvvvpsjjzwySbLttttmpZVWyiuvvLJMLhbTuHHjHHXUURkyZEiefPLJbLrpptl2223L+5s2bZr
evXvnueeey8Ybb7zAv5cvq6ioSMOGDasEi5WVlQu9Wvr/4uWXX84LL7xQZSr27bffnubNm2ezzTZb7GOrex4bNmy
YLbfcMvfee28uv/zy8o+IqVOn5oEHHlhijbvvvnvOOeec/O1vf8uOO+640DaLeg9ZmnPdu3fvDBo0aKHnAwCoXcc
ff3xuv/323HfffWnevHn5j48tW7ZMkyZNUlFRkf79+2fgwIHp0+tgB4Z6e+8bc3pFyiVTE83ZwpJpcmTwRxS2xMxzrNzXU
tt96aKVOmlJdPWnXVVVOvXr1ssMEG2W233fKTn/wkv/nNb5IkRx1VPTbN956kX7+KwK2cdMSTHwWDqeIlEXnYl4qqGNL9SLoN6NDPWKy
cccelR48eOfbYY7Phhhtmzpw5e5ee655/Lb3/423bp1W2S42bx586y55pq57777stNOO6V169ZZZZVVqlxs5OSTTy4
HgdW5wvJ89erVyyGHHJLBgwenRYsW2XfffdOyZcvy/smTJ6d3797p7p27dv1l9//TRv3jyjR4/O8ODDOHDHq1ywpLouvf1T
SJbb50Y9+lNtuuy177LFHTj755Gy55ZZp0KBBxo8fn5EjR2aafffbJ97///WywQb58Y9/nCuvvDINGjTIzjvnJGzvnL
eeim/+MUvyhdGWZQWLVpk++23z+WXX14yR+l6NGjcoNN9yQlVZaaamHPRpLq1FzkZ+JfWazaA/jZjTZFtAVkaXm1KWK7+xXSNRB9rRG+l2vdk2JtWydPBGn1R8Vftylc
ywmO35IorrsiaJCKX6h+8MieffHK+853v5IQTTkkjHjh0zbty4XHPNNXnqqady3nnnnnnpjSHRrwZ5LUHT7pHW67xunEKkz2LI0Kh8pNebL3u/F88++yzOfLII/PDH/4
w7733Xs4666YsvvrqOe6645IkK664YqJ0++ur069cvn3zySfbbb7+0adMMjhDdNmMI35uwGTFA5xtyR28HXrnz2UYOMEMiyn92b3tenagyL1IPx0G6AsXEbR1eBgN0
alDFjxuT6669fYP+vfvWrfOc73l2222XY48Np16dcrUqVPz5ptv5ooEHHsjf/va3fE4A/GZgt7E9Kr8sfeEZHjL7dSjrIfJvIkcgE6dR2+hJIUEcWeRbfVAAzJ/Rqye
vvfdy4YUXZrXVVssbb7zxtWpMkvbt22fvvvfXxOeeedl9VWWy233nprRowYkcsuu6zKxYQWZmnO044UXXpjddtstffr
0yYABAzJ37txcdtlladas2RJHEvfv3z/+MMfss8OSM87IlltumZZ2bU8y6yZSfd+0KMOmd9qrqltjEXmZWhZPlNNuOAQBVo/5jPxvVcRI6OHx2KOgt59dfjYZhWL/dBWbfYvz+S4cAAwyx9RFqPddKkpbe5UuHM/uT4W0pRKn1xQgaLppr2Qk4bDb
110tJSklKI0aMqLLvs88+Kx1zzGljTfeuNSiRYtSjudiIn2p6pnf1uVRWvVdRLtPLU2fPn2x/d5001ZTr2wnzDU2lyS43pwP1LeG6d5jUWjggkxiRoHPrxnaDqnZBK20m0qPyf5U10nGuNaIawoLuOpEuVKyw+odd8rJ8JK20dWfPPNpkqZbtJE34lqe2hfOLjsrK5+fvvkAFR7enYZ4NzTbOpiD2CzvsCxHaxOlq5+WzoZBuLr3wQkQpJ82NJSt1wd9Xq5RHbj6
qrbnmmlXO0ciRI0I0tJSiNHYzi+v9GzLNqStUafPkk0+ENSinxTxXA91O6m4k+NaSs2ZpTg0Kr+nfxZ5tyXhFsaIZYZU8b/9HMsOoYFTwemS2ZJNiNbjIpNGqahk3fEc/UvHTZwb8yIyGzVS6vUFJzcyVhOoaO6Z6KNODd9MSDSunXKZRabNWu6Mm07bPupFPmnyvBN4kKKrWzKu2Cb6GpLGwdKxZ8MtD9sXJ0dXf/g2kH
NWmSbrbfeurTKKqtUafPkk0+WfvCDH5Tatm1btWmGFFUpJSo0bNy79+c9/XuDx1T3P89s9/PDDpYMPRrTsnhxIOPDbOTr61ucnZt4n5VV+l8tRBxjtp8NZYDOfSpiyTwQhMxhwAlTMX5hmPaQmIqUcAl2iYqjMNsVuDHS6a/g61aQ9lYsDzv8GdAVbGMkd1Dnf+cwZ+5YRXn3BdODtokbzFGKOsO
0aVLaY489qpzT+Uang5ZwfxFTnCVfZzDxzt2Q1XIYHzQtSctdW+lgsKkBQDv3hZ1B1m42Bh5WvPUp4ZyxvrcSC8qM7FP3TxzzN2AKhzEdwv0wQFiocJ4NsgjNC5pMP4lboXCq9u6yevUvUjLrFi5vpaoKf8YcXPKGyfzaqkxbdWDxB6HlJQHvWaC1c4N2zVy9GdcmYfxWPyaYNN4kQGRbuFKVgd2Gh
SjBkzFrr/7bffLh1++OOl1VdfvdSgQYNSq1atSt///vdLQ7tsiVIXbC8aauA1W9/h7QoEBf9JyMzQLZ0bavdFFb2Z2sVYpsAkXbV1v4Fk2p3pTxCtB27ZYR3+YrWDnVWaxbSZ/9a8MsM1EcM6Uuq6c/fN3o5PXc+jyXWuu1WGsRVuzwA2GnwoeS29+VA9V2lu6cedxsoZGLtm9ZbYQL9v4tS6K6TZWqW6M+8bK8UNMOW7gVtJG1QjR5MtlNIAklAfG6PzDqZ
kdPzxx1e7vjXXXLO557lv74xz+WNtxww1LhhKu/mvgy+fly+rznksluUqlTVhuXGjZSuFxBSxwTdjpUDLzcmggZJIcWaqM6hkOa4gFcflCvDOd8OD5I+V7QX9/+S7v9PbFWFyOAKDpPT8p3HnLRu/pW2hG7dvsvX2vfhfpsOMrF9B
sWOrYsWPp0ksvXehfPX1WSqVSpMmTSqdfLPTY4d1jd/z1P6NNgDUS5XfU6ERMNQHlbDwCWOlmotwqG2alMeqKqxpmZ2yS3+m8hoxQ6JlvpDDcq3ZKUHCzcjM62TWtvVCb1HSZEWH7v1a+G4t+rvYjU1GGNnCZ+6BLFRtq/vuwQmpebg7blK+YaRLajnMJmV+oN1rXVQoIdDMwbPLnnNmFDbxLo+u1ZQnvgaN+KPPE9uTz4ssGnbUOGdxxhtqrUAUnIn2hSyA/MV80G9rQbpBxIULaqMqawg2hA5Wn4yv44bCwJT9pSDmVBNLtWs6bJeuNXG6yHGKgQmNjeVEtB+4MUkR2njYbRf57y
1XQYAAAAdch7772XNdZY4ntSqVSKisr8+mnny77or4lVlpppbRr126JQbFwM8nkyZOz0kor5b333KuLFi1quuxw0AAAAtGUKVPPSoUOHfPrpp2nZFsm2m3r6I1AMQqdVI6XGb6dBPqaqWa1mLzkUtYIy8ONyDzZ8jDBuqpHf4YLxC5HpKWiycHpGjffMkDFJzd0KxGe3vs1lxeqSY2nkFsjzf81JO5yCLZ7vN2lnBxPMHcaHoupdb8nyAGbLxEarc1LaJpu8RcLkQKjtVjF5eubAHLysX5cbP7Q9RwT6nCg+BFkovuI2dG
AAAAatGUKVPSoUOHfPrpp2nZsmW1Hzd37tzMmTNVb27dCgQYMljticz7T0pJwAt2jRQrgJAAAAQJIs9fTyevX
qVTuUo2a4oBAAAAAAUEjCTQAAAACgkISbAAAAAAEAhCTcBAAAAgEISbgJAAAAABSScBMAAAAAFJJwEwAAAAAUkkjcBAAAAAKSTgJAAAAFJJwEwAAAAAAAUkjcBAAAAAAAUkjcBAAAAAABSScBMAAAAAFJJwEwAAAAAUkjcBAAAAAAABSScBMAAAAAFJJwEwA
AAAAoJOEmAAAAAFBIwk0AAAAAoJOEmAAAAAFBIwk0AAAAAUEjCTQAAAACgkISbAAAAAAEAhCTcBAAAAgEISbgJAAAAABSScBMAAAAAFJJwEwAAAAAUkjcBAAAAAAABSScBMAAAAAFJJwEwAAAAAUkjcBAAAAAAA
AAAAoJOEmAAAAAFBIwk0AAAAAoJOEmAAAAAFBIwk0AAAAAUEjCTQAAAACgkISbAAAAAAEAhCTcBAAAAgEISbgJAAAAABSScBMAAAAAFJJwEwAAAAAUkjcBAAAAAABSScBMAAAAAFJJwEwAAAAAUkjcBAAAAAA

k4SYAAAAAUEjCTQAAAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCT
hJgAAAABQSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOE
mAAAAAFBIwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SY
AAAAAUEjCTQAAAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgA
AAABQSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAA
AAFBIwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAA
AUEjCTQAAAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgAAAAB
QSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAAAAFB
Iwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACikOhNuXnLJJam
oqEj//v3L20qlUs4777y9y0b98+TZo0Sa9evfLyyy9XedysWbNy4oknZpVVVkmzZs2y9957Z/z48cu5egAAAABgeas
T4ebo0aPz29/+/NhtvvHGV7YRMGDcrgwYwYMzZMiQj49ua4u0Pzuf2uxBbOnfPOO/P4449449n2rRp2Wu
vvTJ37tzlfRgAAAAwHJU6+HmtGnGnTctBBB+V3v/tdWrVq1b69VCrl0ksvzdlnn52DDjoopGR7eeq8/Z3nmqTG+ABfHY5z5swDA3eh2N7bXNQKO
3J0kmT56cG264Ib/85S+z8847p3v37rn11sszduzuzYYWLI7V1SAAAADAclDr4ebXxxx+fPfvmFfc9HOV7+//HXvBWpbfFszvb++++8AbbFW6Z5Zmzz
qKyuzyy67lLc1atQo5+ywQb+ywkkyZgxYzJnzpwbKRJKtYsptTWbFrZm9cG8H9RVCssTvOZwWTbMGWJUnGc0tdkMslCo3AAAAAKBY6tfmk995
517/+ldGjRy++tkbRSUuUt+7Ky+zyl3tMRWLS/ydoiyqBJlQ3kl9lmz8zmQQdlTtmJi5TEWO+uBbDFPqTqOnpl2K3Sd4Jr1Zrsmmd8+9R+9XuLF6Lf
eeznn55JNz/0k3vs7rn11ahvivVvGfzqGGtpz+8PRYx5ee9xNXQfWmyzP+atf1nc+Ee1QBHV6GlpmaIb0y3b4
I9yWmz9JejgJ2Ev0RAfKWGFKeCdM7mbmrMmWRCeP4a2+GHi9nvCHtbRqLW1rt1svMmEVpvc/mgEo1t7ur5jOgJwgg+avTtA8QktXxEeTGvc
afY+5/JCv3QcAAAAAy8+isiLqnlpbcxMAAAAA4OsQbgIAAAAAhSTcBAAAAAAKSbgJAAAAAAKSbgJAAAAAAKCThJgA
AAABQSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAA
AAFBIwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAA
AUEjCTQAAAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgAAAAB
QSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAAAAFB
Iwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAAAUEj
CTQAAAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgAAAABQSMJ
NAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAAAAFBIwk0
AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAAAUEjCTQA
AAACgkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgAAAABQSMJNAAA
AAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAAAoJOEmAAAAAFBIwk0AAAA
AoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAAAUEjCTQAAAAC
gkISbAAAAAEAhCTcBAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAAAABSScBMAAAAAKCThJgAAAABQSMJNAAAAAKC
QhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAAqpVsPNX//61PNX//619l4l4l4443TokWJ2GjRIj3cVcnjJ1+/nv/XRj2N13IDXIJEKJPViNXT1U3l8qlXL
eeeelffv2adKyreTanGf4HdOY3gPQQU3xnZW8zbTpkwPEp+6eh13r5xuYDWhO++sK1kFlc+xfjj3sXpUPhm4dbLT4NqNOrEH7Vy8n3G+nKrhgheoxG/
vvfbK3Llza+uwUZL3OLpkMHeX93RD8vwLcGxA2gWqUwYMHZ8iQRk9enTtawuWnDkrOPn36ZPxJ86dgWI7zEjnKLWBcDHtWdvTeXXEb4NKmb5/8Tyt6SMdn+Tim+M+w
58uSsuuqq+f3vf58f58DDjggSfL++nQ4cO+ctf//pJd921Ws85ZcqUTKWc8ZmAfJD/fOafwQ9PGU9M89FGzuZMiZ2ap9xs+k7rTcXH/IPA1+ct4KkJmOsSeoRn7i+QXbPNtp/ytfs
AAAAAYPlZVFZE3VNWvZS3nZW7OAy+lLoBSLVBtHRzEzTz72lfu1b3Nk0lhYi9M59K3bdcvmgtbzrRDA8jVn55W9e5kcDkBx4ZbdWh43VVM++v3XwhGyFlz6BzUktXqN6i/sTkIfvmKdqLtWU0cfHIRwBfnWZPbKhCltOm9+UBi5LHW3GSx1ne2
r6VoAAAAAAKqtWldL/6q33nor//d//RWRnOtHzV2GaAOTGA0aFRWyb2DyuRqGX9xQUJA5a+o
zzz2ZNm1akuTFF/MueeeW+MFgRaSj2eccUYx7GqKaLQ4AAAAAYFGWOtw7927d5ud566qzavkvUiAAAAABWMzDdWuGMN3h/5sEjelb0bvhmBDrKe24VJVRtrmlvW9WJXXYeswNB1wrq3UWUyTY3+7QpK9+nUQY+7eTXBlTXoUN06uw7rK4t

cO3Zsvv/97y+wfdVVV83HH39cI0UBAAAAACzJUoebK620UiZMmLDA9ueeey6rr756jRQFAAAAALAkSx1u9u3bNz/
72c9SWVmZioqKzJs3L0888UROPfXUHHLIIcuiRgAAAACABSx1uHnxxRenY8eOWX311TNt2rR07do122+/fbbZZpv
83//937KoEQAAAABgAfWX9gENGjTIbbfdlgsvvDD/+te/Mm/evHTv3j1dunZZZFvUBAAAAACzUUo/cvOCCCzJjxoy
stdZa2W+//bL//vunS5cumTlzZi644IJlUSMAAAAAwAKWOtw8//zpAbeIsw/gxzjI3IRWgxqmWpeOg5xDBMRe3NqQ0WUTzEMoNDzV9r/G9/LaFATJA5jkPPv7RFpqdkJxgBmlNjpJhGkMSXwh0+nPfqMWNRn1f7xnwRbNCHLM+nvzOZ7uNbVBC1jiGlcWAnHa29SfWu6k4nhcMI5eAdmG
uu+xS0/UAAAAAAAFRLtcLNq666KkcddVQaN26cq666arFtzrppBopDAAAAABgcaoVbl5xxRU56KCD0rhx41xxxRW
LbFdRUSHcBAAAAACWi2qFm2+//fZC2+5csYOq3eHgwYP/52IAAAAAAAKqrWuH
mc889V+X+mDFjMnfu3Ky33npktdffz316tVLjx49ar5CAAAAAAICFqFa4OXLkyIJ/Dx48OM2bN8+NN9+Dd
mTZqUww47LNttt92yqRIAAAAA4CuWes3NX/7ylj7nkkvKwEd8+fnzGy+D4IBJhatAEM8xzx4W4xgS9HpXZ4XNM/YoHgEj9h/F4MF5uvaqqP/A6B80Wh2wWcBFkoSYBXHPVlhAZb8YXfQFzFVXGsMq2ftwA7hgvIy0GvFCqTtd3J3B1e6bkWaF6jgc8Qxj/lvc/j/p/fKHOLKo1/lvnzY6QSXChbRPQzHFZ2CG88lUeaSNlgAFBv5hHfIAHRVVF5N/9rHd1WIaR8L4ZeI9aQp+OJ9mbtQ/ls33AXcrufiErlxHkLaG32AAUHLAi2QpVetQ1sQrCa+ZGZvHc1mO6YWYUeN9TJdQWgZwf3p9vfqMLfTl7GmoL2d1mf8zdfYJ+AbGzFI2TmnMfa4zO5Xpdc2qY6OuhRzY5DfnrcvLFMA7QuuzhTnF3sxrf8Lh2Tt8JZ8w+Ub0zsh7/4YoQ+NCfw5aLDAiBHP+y8T2ZTxetKBCDDdcs2hsmgWuAmzchi5aOJ9KNQn4F6I5lhxfy1NZjIXmD5D5Jh36m6jETKb3jgo9kkVuC5Yd8Kjdh+1RSq2BBnw77bjIEn8wLq2/YuFBcOo+V1w6kDj2mXT4r6Ssv2V6b0qhpUfzUBlz5Q1TDcZuAZ6w55fvPZyXAqNGIgQS1zNEtlaxOW9LxRMVsbMMOcB7IfQzGIH6a5Sj9T1Jq4f+20z9x3jO+rEwBn9S9MRuPyhC7aJJ95xuTyevtfLcfaUUC+W8K2XwlhGgj2a2ZSCdN2hSVsBH7n2dZQOV1xK5CZaLmVN7ZYjlUUT4GKbm6hOPKWM/4gqDDpjIOC61Kw+T59IpEbPOYNhSeP0H/OWBPzTlHPdS3Xm0ZzcTvUIzCNfPOEF5pCnsgg2+hvOlVzw8L4qsFc1Wg/k3b7wbnqWVnMdFmZN/ZkUj/zOhmrU3pPXuEjhS4OL0MYVddvmX+RCpX6GB+vYD2OVoRStT5lq/FJyFa3QggpOK/cF0qrczzxVKOpQ5ZqpN3Ov5TMtMe3KrKbTLS3vA+eJMyfO5ZXFbwzNwwLFmqmQDQcewhYAy4o8CMSo1zUiSvSaGzTaUOzJ1+5h11jl7Ex2nCxcnkB5BmzR8+ciULmw57lhF5Wsp3xUzS7RrYHP8TBGz8L4nL4+Z4rSpt39JR5Y4Id7zE1b+0IJ4YuwwZzKqUfw85hy+0YbgS/MFmYqkDNYv1b0yTmzD43/rtRR0+BLGmbxRLB6T6WXMLlbKyR2q7D3BaUqUfcA0/1PBOgxN08bGuOzLFUp/
cO3Zsvv/97y+wfdVVV83HH39cI0UBAAAAACzJUoebK620UiZMmLDA9ueeey6rr756jRQFAAAAALAkSx1u9u3bNz/
72c9SWVmZioqKzJs3L0888UROPfXUHHLIIcuiRgAAAACABSx1uHnxxRenY8eOWX311TNt2rR07do122+/fbbZZpv
83//937KoEQAAAABgAfWX9gENGjTIbbfdlgsvvDD/+te/Mm/evHTv3j1dunZZZFvUBAAAAACzUUo/cvOCCCzJjxoy
stdZa2W+//bL//vunS5cumTlzZi644IJlUSMAAAAAwAKWOtw8//zzM23atatW8NFAUAAAAAAsCRLHW6
WSqVUVFQssP2FF15I69ata6QoAAAAAIAlqfaam61atUpUpFRUUqKiqy7rrrVgk4586dm2nTpnTpuWYY45ZJkUCAAAAAHx
VtcPNK6+8+8MqVSKYcffnjOP//8tGzZMm5Zsyt1zX7sa1Z4IQsQWRGuAPN1l+lrZyf7HB9FTwS8gJMZx4ARi3tPpqW/aKq5s24L1Je+8I4UHCAl7TGQl0iVNQNxLHk+sDRUn4B
VtcPNK6+8MqVSKYcffnjOP//8tGzZsryvYcOG6dSpU3r27LlMigQQAAAAA+Kpqh5v9VLvVKfFJyhOjyoHK3o7NxMZPuBjxGr57mAAAAAZ9xDrrt6Fo9NTD9f6bu4drs9PTgLBf54h4JF7aFaKG+kNJ6V
tAwAAAADUmKVec3P69Ol59NFHF9j+i/+NQ899FCNFAUAAAAAsCRLHW6eccYZmTt37gLbS6VSzjjjjjjBopCgAAAAB
gSZY63HzjjTfStWvXBbavv/76efPPP2uk0KAAAAAACAJVnqcLNly5b5d3/+s8D2N998M00bNqyRogAAAAAA3f8fS26BNf7+aSh1ULyDNdMtV/TVtxxm3LuMXt8MqN6A9SmLqb2T0AoB/2rplYxq5cGEPFaxAH/aS7/zLt2
99947/fv3z1tvbbzGa2bZjz3BHQPKnVYb7+n7SwXe9eVXXX16i5QcCCI+mYKzDc34GYs0wK5npWDqvqOMlFwXtcG5dW6gO9m/A5tKTzbcDbEC1N4/5X9I
FL36xLGoEAAAAAFhA/aV9QMuWLfPk09OmxIgReeeG99KkSZEo0CCAyyZtHk4MfW09nuGwG9vV7lMjUb9ZmvT+5/o2X6c6ovg6cSjqDv2rhTGpOdOTp2aJftNmKW1BBPxbAXERmcjjovg5eDh6wR2ZGKkL7kX
uu+xS0/UAAAAAAAFRLtcLNq666KkcddVQaN26cq666arFtzrppBopDAAAAABgcaoVbl5xxRU56KCD0rhx41xxxRW
LbFdRUSHcBAAAAACWi2qFm2+//fZC2+5csYOq3eHgwYP/52IAAAAAAKqrWuH
mc889V+X+mDFjMnfu3Ky33npJktdffz316tVLjx49ar5CAAAAAICFqFa4OXLkyIJ/Dx48OM2bN8+NN9+cVq1aJUk
mTZqUww47LNttt92yqRIAAAAA4CuWes3NX/7ylj7nkkvKwEd8+fnzGy+D4IBJhatAEM8xzx4W4xgS9HpXZ4XNM/YoHgEj9h/F4MF5uvaqqP/A6B80Wh2wWcBFkoSYBXHPVlhAZb8YXfQFzFVXGsMq2ftwA7hgvIy0GvFCqTtd3J3B1e6bkWaF6jgc8Qxj/lvc/j/p/fKHOLKo1/lvnzY6QSXChbRPQzHFZ2CG88lUeaSNlgAFBv5hHfIAHRVVF5N/9rHd1WIaR8L4ZeI9aQp+OJ9mbtQ/ls33AXcrufiErlxHkLaG32AAUHLAi2QpVetQ1sQrCa+ZGZvHc1mO6YWYUeN9TJdQWgZwf3p9vfqMLfTl7GmoL2d1mf8zdfYJ+AbGzFI2TmnMfa4zO5Xpdc2qY6OuhRzY5DfnrcvLFMA7QuuzhTnF3sxrf8Lh2Tt8JZ8w+Ub0zsh7/4YoQ+NCfw5aLDAiBHP+y8T2ZTxetKBCDDdcs2hsmgWuAmzchi5aOJ9KNQn4F6I5lhxfy1NZjIXmD5D5Jh36m6jETKb3jgo9kkVuC5Yd8Kjdh+1RSq2BBnw77bjIEn8wLq2/YuFBcOo+V1w6kDj2mXT4r6Ssv2V6b0qhpUfzUBlz5Q1TDcZuAZ6w55fvPZyXAqNGIgQS1zNEtlaxOW9LxRMVsbMMOcB7IfQzGIH6a5Sj9T1Jq4f+20z9x3jO+rEwBn9S9MRuPyhC7aJJ95xuTyevtfLcfaUUC+W8K2XwlhGgj2a2ZSCdN2hSVsBH7n2dZQOV1xK5CZaLmVN7ZYjlUUT4GKbm6hOPKWM/4gqDDpjIOC61Kw+T59IpEbPOYNhSeP0H/OWBPzTlHPdS3Xm0ZzcTvUIzCNfPOEF5pCnsgg2+hvOlVzw8L4qsFc1Wg/k3b7wbnqWVnMdFmZN/ZkUj/zOhmrU3pPXuEjhS4OL0MYVddvmX+RCpX6GB+vYD2OVoRStT5lq/FJyFa3QggpOK/cF0qrczzxVKOpQ5ZqpN3Ov5TMtMe3KrKbTLS3vA+eJMyfO5ZXFbwzNwwLFmqmQDQcewhYAy4o8CMSo1zUiSvSaGzTaUOzJ1+5h11jl7Ex2nCxcnkB5BmzR8+ciULmw57lhF5Wsp3xUzS7RrYHP8TBGz8L4nL4+Z4rSpt39JR5Y4Id7zE1b+0IJ4YuwwZzKqUfw85hy+0YbgS/MFmYqkDNYv1b0yTmzD43/rtRR0+BLGmbxRLB6T6WXMLlbKyR2q7D3BaUqUfcA0/1PBOgxN08bGuOzLFUp/

729/SuXPnKvs7d+6cdu3aZcSIEeVts2fPzqhRo7LNNtskSXr06JEGDRpUaTNhwoS89NJL5TYAAAAAwDdPrV4t/fj
jj8/tt9+e++67L82bNy+P0GzZsmWaNGmSioqK9O/fPwMHDkyXLl3SpUuXDBw4ME2bNk3fvn3LbY844ogMGDAgK6+
8clq3bp1TTz01G220Ufnq6Q6QAAAADAN0+thpu//vWvkyS9evWqsv2mm27KoYcemiQ5/fTTTM3PmzBx33HGZNGlSttp
qqzz88MNp3rx5uf0VV1yR+vXrZ//998Mm07047ZejdQoalXr97yOhQAAAAAYDmrKJVKpdouorZNmTIlLVu2zOT
Jk6usv9njtFu+dt9jLj/ka/cBAMtLTXz2JT7/AAAotkVlRdQ9deKCQgAAAAAAS0u4CQAAAAAUknATAAAAAACgk4SY
AAAAAUEi1erV0AAAAvjlcmA6A5c3ITQAAAACgkISbAAAAAAEAhCTcBAAAAAgEISbgIAAAAAhSTcBAAAAAAKSbgJAAA
AABSScBMAAAAAKCThJgAAAABQSMJNAAAAAAKCQ6td2Ad904y7YqEb66XjO2BrpBwAAAAC+KYzcBAAAAAAKSbgJAAA
AABSScBMAAAAAKCThJgAAAABQSMJNAAAAAAKCQhJsAAAAAQCEJNwEAAACAQqpf2wUAAAAA1dPjtFtqpJ8xlx9SI/0
A1DYjNwEAAACAQhJuAgAAAAFJJwEwAAAAAoJOEmAAAAAFBIwk0AAAAAoJCEmwAAAABAIQk3AQAAAIBCEm4CAAAAAIVUv7YLAPiqHd8kg/ALPBIwkAAAAAoJCEmwAAAABAIblaOgBQ48ZdsNH
X7qPjOWNroBIAAOCbzMhNAAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwA
AAAoJOEmAAAAAFBI9Wu7AAAAAPiycRdsVCP9dDxDmAAAAAFBIwk0AAAAAoJCEmwAAAABAIQk3AQA
AAIBCEm4CAAAAAIUk3AQAAAACkm4CQAAAAAUkntAAA
AACgka24CABSQ9Sv5pvBaBgC+DiM3AQAAAIBCEm4CAAAAAIUk3AQAAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAA
AUEj1a7sAAAAAYPkad8FGNdJPx3PG1kg/AP8rIzcBAAAAAKSbgJAAA
XSDcAAADwrWDkJgAAAABQSMJNAAAAAKCQhJsAAAAAQCEJNwEAAACAQhJuAgAAAACFJNwEAAAAAApJuAkAAAAAFJJ
wEwAAAAoJOEmAAAAAFBIwk0AAAAAoJCEmwAAAABAIdWv7QIAAAAAvonGXbjJ47l4zEf6UXKIyQ1W8BHKJOFFMSR4DpNXs4XwZLBIMvLf7WasD5Xh+w
baqSfYc1rpBsA4FtKuAkAALWspoLCMcfUiBNqAAAUhXATAAAAAAopCEm4CAAAAAIUk3AQAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAA
EAAAAAArJtHTgG2vcBRvVSD8dzxhbI/0AAAAAxNFIwkOAAAAAAUUhBQAAAAAQk0AAAAAoJCEmwAAAABAIQk3AQAAAACFJNWEAAAAAApJuAkAAAAAFJJwEwAAAABgk3AQAAAACkm4CQAAAAAUknATAAAAACgk4SYAAAAAUEjCTQAAAACgk4SYAAAA
Iwk0AAAAAoJCEmwAAAABAIQk3AQAAAACFJNwEAAAAAApJuAkAAAAAFJJwEwAAAACAoJCEmwAAAABAIQk3AQAAAAFJJ5JUq+u99N+/+
p3759mjRpkl69euXll1+u0mbWjWrFk58cQTs8oqq6RZs2bZ+9M9pN378+OV4FAAAAABAbajVcHP69nnpZu23YTIkX
uHzRoUAYPHpwQ4Zk9OjRadeuXfr06ZOpU6eW2/Tv3z933313RowYkSeffDKff/559t5778yfP7/c5m9nH+zmrJfZgPtt
92nzy3Xff7vPaUAJZ//U7+6+f/RwngDBRkw5qs6+R1+yCXnSDrydGY9LSgKIpFI0OJ8SCJ7NzXJNAAtN+jAF1GRb5dkaIiccKn5CcrlOtHlPbRlxtRlpt+7G9E6mSBWkdWgcl7tGpZ2e+JLgOg5TXpkYRVpGuDcjKb5v7ygGCTwNNCqaTwBQU7DFmfUffnrVj6iNmQ1ITAfGUTmvNp2ovNHhcr
zqMZ0O2E0TqJgbnnaw9RzDnJ7Bf/kRvKyF5BnXGN2F9bBDzWpNeVQbk49U9nnZVyZQ+G4RXnSxJ99Vc6Up6WGu6SfusDJNDi3X2lxJ0FRPl8vf2T6rfRqTMeOomvmdyyTTf1w4gWxQzqyRdYM2BVhdqSUyG6AI
KWN25+ZnmrIfVkBH6B9JSbY23cU0F6tPYKMPuvnKn9h1tgWTbMlUf3j+bP/kvAp1m//3nvzqm9qbdXdVGNAvLxAQo8pmXSMdsHKWXJTTqP/C3OpKujY/RKi6oGbLOQuX+nfX
0jbQm9uytlFsfIDEwXV9nkS+1CsZJuXpQp2KfTrm0f3NTQ9AV/bhNX05vpPjCNpBKzgGhF+z9OVbEVpYdKeGL9r9ZqS6EFtPFa0+I+1Y
AMuGBJhtT9wjmI5WnFc0TY1ibdmdRHpUWgrPbMaKe0k6Gqn4kRsxW/zvc0+UkdVbkKDBMA2A3xVjm2VmpVjN+EP4V5aZkUUwz9h8FCbbuCZGDkZpVPvlL9V7FI8ZLtzq29nU43c+bnFMWVUOKdD3SMDaQLx
GbVRmt1b1V0ESCdtqeB9IN0HSK9+1Q3CHoyI1/ehgjPF8j/9G4Uq9jajO4VmoUiCkj+T2M+xgI1wbgSEBuFTwLPyURTSPHpqh9GCgb
YkLPfRGgNyeQv1MIJtyOU7YM/HCk0cNNH4xvLCbDEJ7rJ0iddOvgNh/1Af+RsdF8EdmnrMgJlRcUZe+d6k1hmW/JNbZY3SVyd9jvwvzsl9nzh21jzv3fLzs3+33hnMKDRJ/4Sgsk3+l3F7rlYrWYpF3
pMoMlHyAnf8/1Tk13z+hZ67PX/5j3W8QlK30jfsq1NO0H6MnVPeJ3/5gh+bsmWW15zjH8ZxYRyHvHq2kOxlp0Kjnrbo/TTbOm+CUlKbtYNOj08yAa35GXoHLF/XCGO5t+bz+TVnoZc7oRTMG/2IiJ4bNp6
+ICAhsOrDxeZa2lrIOrpOYYaaffUW5LGUVkiZ0hHpLSsZ1Bo95HdnYQmjpmnmGhNkmoNC/95rbvTeFgFF8B3K7wNYdMtgSdjjlqtAiw5yT4ZuNNMqnMVvpRvD/1hrw6jM+OK4Gh6PJEwn2uYvlJThT9OsP2qJD/
+uqJFhu8zjLWyrDnXtl9+RZptT5ogXaaC6NPtefv9NVNqyzwN7E4n/bRJ8wJPB+W/kKYejEe0PIWlgD//GtYxjYJp/DkDjG+/+Rz99k+eW7i+VWSp3AUMKN5HMnDKxNRSC9fA5iXD65lROfPAZNXcWvv0ec/4+
Ac8OVMUbRd5S/HOWWxjfH9hN+rLd2lgCNqVp2TSlvG/lnCQ/+xz33zhUYzMAu3KsE/3CUYFuyfKlXRBObdK+PXBU+1bJzQBNLt4VcnhWqs/WQ/ZWBWwNqsZ6wXm0Me8ESPdd5Nw8vWBcP/PYy61W3z8
QGjzCZzsj9mGrubB7aT3QO6R8ClBwWJfaaI+9TQeXxlOhw89a3qHl2c15b7MiErZOv67oK7q2PQVK5HKU2EsQ/QdAvDbHXJ6JeFzv1GC/lbPWJ7eQS1VmHFlmfRqDA1PNlg3bO+nOhR5cj0OeSh9pn4FqTJ4
8zV35ai1RBf2WNeRoD8ruhVJOZvgNHH3qqRvPnF/VfO+tIdWIHAu9oKJv3+AnWBd0mEr0oOv3Rv7DyhL+qFFtRvg1qYezKUM+0+1JKuaPWFBYNW1hmkf0ubdxn4EHDZEJRHamE7P+OEnkq+MjzB
bhzXYgMRqP2LUqMPR+3CvDWoTsu8pQPNOJ2Vbzj2jfWzg1o8Kdx++1mfTNHk4XZhwQ7evA4+NoGvVO3vhm7G/fnDg72/hv6xDQW6R1dDxKDjSqcMbLY6nT6NtYeL2fNP3Ct9BjjeTZ9wEt+OcBNbL
vbsBsgK9G8PR7lTSTeKmTMCzM/LVvA3cCMvbXjt/f/QyHRQHZ4ZtCyvNOIn4gPGjDhAy4LXZ8sG7wk8z3EeRCy91WpkIIPhxM03HqYpdxh1GPpI9TL3QN3mZePyBCCrWXIbbk0W5bSVv3nHyOlNTRfk
/z3hVxz4nkXHvTERCtlHpPFeOsmL0Fd6Y2aOtSJ7hOO2G/KUEgk5GXlDtkl79Pt2lbpfOmdy+g3VfMrAFS5O2aOkM5RX0lswr43b9vzW9n0s0Hk9kVVx33kJUVmHNdtCHI6WOSEGLBzqsDgK/VdK0pA0
PvBI+Z1hSlB7qSq1TrvscC0MqdURkzrVDgevMkl0RVYKxO0mvsQNrlpPEXxOgPRy17nrFaXdkyfBT2m

zz1+eZQIAAABQMO+9917WWGON2i6DxSj8yM35KioqqtwvlUoLbJvvzDPPzCmnnFK+P2/evHzyySdZeeWVF/mY/8W
UKVPSoUOHvPfee2nRokWN9bssqXn5KFrNRas3UfPyULR6EzUvL0WruWj1JmpeHopWb6Lm5aFo9SZqXl6KVnPR6k3
UvDwUrd5k2dZcKpUyderUtG/fvkb7peYVPtxcZZVVUq9evRVbZPnixLLRt23ahj2nUqFEaNWpUZdtKK62OrEp
MixYtCvPGMJ+al4+i1Vy0ehM1Lw9FqzdR8/JStJqLVm+i5uWhaPUmal4eilZvoublpWg1F63eRM3LQ9HqTZZdzS1
btqzxPql5hb+gUMOGDdOjR4+MGDGiyvYRI0akd+/eC33OjBkzUle3aou2/qu2/qu2/qu2/qu2/qu2/qu2/qu2/
lmGOOqe3SAAAAAIBl5BsRbh5wwAH5+OOPc8EFF2TChAnp1q1b/vKXv2TNNdes1boaNWpUZ3eydtPm8bc8yGnSeXk8Aq
ctHoTNS8PRas3UfPyUrSai1Zvoublpo8wHbYumpaHopWb6Lm5aVoNRet3kTNy0PR6k2KWTM1r/BXSwcAAAAAvp0Kv+Y
mAAAAAPDtJNwEAAAAAApJuAkAAAAAFJJwEwAAWKY+//znH/++XnvvfdquxZ9FPQe4SYsY3PmzMlhhx8W//z
nP7VcGCwDwDTBnzpystdaeeeWVV2q7lX+Lr/88X9999XnvfdquxZ9FPQe4SYsY3PmzMlhhx8W//z
aNGiQYCoG5eyzz67tUs3L2+++WYmPm3vdm3/fbb11JV3zyff//55Hnvssbz/55Hnvssbz11lvp27dvmjdvnvffff2
tWrTIiiuuWNvlAbWoQYMkq7lkWy884757lKWy884757HHHsuhhx5a26UHX/7/7+OOPc84552TkyJEL/W7/W7/ySe
f1FJl1TNt2rQFam7RokUvRbSfOfQQQ2XQcN4ckT/Qu1jNDa1S23T1ue/qW8/s0g/7w6WkvBoePfQQ+k3q0aep7f0bdXKqbjwbU9QFiTlvwF
aLfZ818U3409///v505/+L//+17hSbhzzqAjJxvDeSQOBIhJ8dwVbd3kTNFkRjtRu//O/n0e7+gBZFyZUE9j1AqSs/vAXnX2j9Nh9fzdjvsvx5DA
33DKlUimjR4/O8OHDc/zxcn+88tO+9Rhf25zsyfl84J9ei6vGaw2+ByP2J9Ie0+45j13iyhqG0XnYfhnSrj1nbJNkC/RRKD9YPPHbcq/iwc8pSGgcq9R1sJU5nvjSWF4hf4w8PP1NzumJqo9Nsbj13QX1ns
CBLIsexWlrw694Ws58sgjs9pqq+XCCy+ssv3cc8+Nebcc8/dbNNsdb9xf5DH+ahTHzSdaDl1esCNh6Vi+Uxz2tAnqKAHZd+0GfJrMxRxGg8VVEQ+k+47zsKm4BC0Ft2eWhdehPrsgB4Mqo+Fz
HOXny5Cr358yZk+eeey5nn312Lr744ohrmaGvgUVRGQPvjkXZkgV+3iVYn0bHa9n1WKCrWC/aVK/EXgc91qzaqO8PdG7I0lSKV0qKAvGrzuHsIvNwL7m8UoPRqY6Hy0m5gvQNLTSNSaR8ldyfYbfEvnAbWWhIYc1rYdJ2
aTpV8c51//vk57bTT0rRp05x//vHtyvcvaRp05x//vmLbXvuuecup6qq7/jjj8IkSNzwQUX5JBDDsk111yTD//73Y//73v/nNb36Sy+9TSy+9tBw
Q1EUffPBBRo4cmQcffDB/+MMm/evDoXChVV0V4XP/jBD9KnT58cc8wxMPP5x77rknV199dX77299/ycMPP5x77rknV199dX77299
m7NixtVRlcW2xxRa5+uqrs/XWdW4YDJMWcQF8XdW2SDD1pESNatGiRen1119fYPVrr79eatGiRSM1UTThTF
o9ezZs3TmmWeWHWYtHtHH31UWmGFQa+GFQeIoNGlTbbbLPaLuMuMbYdNNNy2tuOK
KpUaNGpXWVXXXfdUvfu3avHbbbbxvSrbfeWouVwdLp0KFDaeT
IkaVSqVRq3rx56i5YY033iiVSqXSVfMVNSSCk3WzgagXsDOnkMVQ7NZNNDuJUfVrGB9wPtRSq0SsRh
K9rpo1qxZucYve+ONN0rNmjWrlZl8ZpqwTMvKwWRbfNbp2pgU2wWlOtVZMAVbeVhHRGwH3xmkOjZLRTJcgT
tLurVq1dpxIgRtV0GdxSZMmefzxx90lS5cq2x9//////ec//9nSwqwlPl9xL9FUbAXtlZUb7IM/Zy6UrHrM4
NytM/dt9999oucQGlRQw2njVrVho2bLicq/l2WnXXVfPaa6/lM6okZMGBAKis
rs9FGG6VBgwZV2m688cKbLu7xq+eMf/5i77ror48aNy+zZs6tC6KaFZs6v+3tb5N8MfJx2rRpOffcc7P
HHnvUcnXfHOuvvv35mzpxZ22VU20/VU209/+tNcc801/+Ul2mGNdRdVK2oM/NCvH5M2o3ozfp+HhNOLvvb98frbgBw3vv/DNeRVPG8X1Dx5fzG5eU
NRTN9+vSMGjVqoe/HdWXK/+LMn7octYOzYsZYqWrRPVmsWW5VOv5YxzxjVSzNzg5cVtYX6I69Y
tvXr1ys9//vNst9125RE4l1tpzG9i15ccWfyFlJXfRKZHRRC0hsTRU7czGCjgfbLuLfGvetiR9S1SvwgN6djazawF2SjLRiAJyTWkrb1UDRp5wvQ+ZiO8V6yNsWeTSd0bnVJ08vUfHeeMv+hOOt70f29XKVoZdwnvvBQCmHqaYEyjBZrq1W5bMrW8RQPYN6bQGHx2ZSSMSQwKQGH1zt6e75ziNSKIM7Qmt2jTuTmO7iPetsaEFnmxOGx+O9XoaMHQcxXrYHC4/qmrTQoZQAQaZPg1STLoQwQjFMo4WiZ2MxW14zUQLQY6pJKNmHDA6eNnAEo2LNUzQoc2PZdSO2H7BrntBz4xB2ksnA8NQ8HF+CBzHnrhTZpxs4D+eU1oU9jo3FMk+EXb6Ghg2MDUkdNgOPwTnNVxCaf+6p7QnV2S3aJZQnGhbNNW9rLu5c15JWj3T5e6vW6jCF5Lgq0n77cHR+HrkfqtnsZ7+eqMK1Qe6/W7+tHtq5xOHLrnqHIWJfFX8mj4kwyKnbM+8oXZE1BbPy+UXpxvFMvD5OmrnD2oTkpCLuXyf++zxn8dG12yycmk5iR2+HyNnf5iDBTmRTWN9HvTA0NW5c18gKoxyI28krR5AQaN45YDqdvz8XUqXmpHF0v6bWlBO6lVzkpK8LPE4p1XrAsPL9H3t0LLbnhKAdl0uQ1EgNPIa8N4ZxMFvs6o0ZAGdwIP2Yz9M7ZzupaKZpEXz6ePdXzX8Xj1CArvN4S4bXg1PyCzt4Tgn6ETaWw1p6zZvpZ9/dD9fsqT7k1hTP72XfXWm5gG5fHZzC8Kr6MH7lSr2Xm2umsjN2fnxPl8N7UHcw9dNJZSGPqEYFaDqDbGd6AVO4X6uqOqbEz89jfP/+TVKJSvQm04VcbS7znBVkkvHyFbz5Wq8/nrzzYQm9bT4+i43wXVLtc//Jy8hvvCJhhTVc1dvJKr5uhf5UFyOhq9b64zWBKKGbv5RyLQeqJDnvvz0SNsfwzjJ3QE2lgGP/PQHMxD99PZA3Qh6jvMcbgjvTc6XCz4iN9SPnUoXXKfbkvoyoz72CCY1u0ktqsgTLoQ+frO7ggtrY1HeuTySCk8UUbtXVXe8Sxw2qJxehZnx3OsTWyxXlYXJhFfBXZFknw+KrPVRWRxBIsEmpOzaHCgyDdw7ttHdC+8PdUUKMdRmz7KpWT9tsg3qPgKL5xFeR5dZ1MfUfuQsVLvsXNn4Uq2PB3cDKLbWKvQc2CRsnmVKoyWnSP0KgCVHSnFaWmf9swvU+BWU6EM+Nl4W0g3eqGRlrPgZG0Tf4g9cKnSPQ9D9bfvF3qlcQXt5x0IC0X9+4jMb7tWTnwvbJUCQI+7QvEwD4KKpgaw+9NQYPvN++qmzMWEQZ/fGMLyOUDe+qxO6aVYGvMr9l6hHSHG7ZKPqcQ6l+rPxUQeI3uYpnkj2++8h8lOgbXxSfR4CMTrl81ksLbDMzmBsW9hZkkJaKkQzUYw3tIFhBLCzCBOk3oJRXzlbT/nhuC79kvsIVP2B2Nq1BDnufiDg5WZmPNQ6aDMfAeCb2YcvQtdQzJyVPJVIeg7DjGOkNuFS1Nl/HjBvYZ8l9zlEUFAMhI0DdJhB7NzsAOQOljdP3ZDsp/XWvrFh/DC4SLo/JdH89y8+fAfVXknPzm+U0LsUOPlaVa9q6jGnKfEqEbdzyRbg6uaLUG/JCmdDSqvfVsWIRj5b3GmRW0bU0QdIVzLqwoUOG9OOkVfCx+vA//Xwp0zuFkBHAazeG24YKzggWOJSGYqVnNX6OcTnhFf/bgzvwvw/GzWCy0Zq2GSlGiSwD4H3+XRORiOjdyTxXNXM1Nl2DBkRXDdFHQSTbUVIRzL9UCbRLsf1dqSOLwEXVK0RfYzPGbhJyrQKIU3MaUVdkVabFjWGZsTNe2d9KH7yLGD2ZDNmEYa6EvKUaE3N5K7gLnZfKLR8I8Pdk4Vg1U+bPBHt9F4bz4iy1KSCrPY0vWRDXKlNfyHBvvjpVF9x0C+HZ0qPFwxYHz/CVYKkDCn5/S1yDBSL1HHv2fK6VX1vDw+/yWfeuRvKrMGhHmnLbD3tPjRa5NKZnJOpqWR7BjfWNnddZWtrTw+2vbfHDfhz1+yVy0HMB4Bh8dgZhwq1z4bVvA6wPxmTSFqrvAlt6DN7c+2+AiX81R6u9aPUHSt+s+HRcJzfKPh95JwYE/b12F+XYdBXS7Pv+w/vuxAUS8gzk8SIU/J9x2KqBPBm7X2W+wuaYCj4XTq1PRbtFkOoU7Kx8w0hDg01PEd7Ac2xEVNP9oSb9fKwK2UHYh/Z6A+mG9zEBTvvw5UQhjeJzd8/ugVEgPeUoYIB2y76AD4L9ADFBEhSKOcfnZ5Y4ieWFKtB6YWtpWfbZfTJ+Xl2fT26zdeCfDZcw+OTtgAq3bJ8//Ba7ykqnRzH1VjbhxoWTqBKlYq6HtxcbXUEdZPYmdtc0ck0AoDUYC0aWQYqrGnHC/nXeBQCNVqRyhz44B7Bz9CEYDmzHcAKRXLi3IASuXKcqG2Yv6VvVe+EQqs7DlMJ5LHSxRLqu43F0Uf9w0nQu6bdIjBuhQmhv6mBO0dKoX+gjPbm5wIDKIq1J3C+NOH55Ja1r3zzRjoSQfJ0SDdP2iGtVqb8wpaRoMx5bDf7d9Ju+L/I4mD/2VKXN70D1jgFmL9RMdvqWX4Dhq1YLemkWzb8rhiC8P3WT2mxAvyqwE6GEfJyEqsGY46Qq5TC+e3Pxd2d+7kdzyZxkMP8B2P0Dt3Eqf/1w0VgkRKLt71ZD0zXQrEXrTXU2K1e5HzJcBnf0T36yQVX53nEeYtqfXwMtkGkECsbJWBG4sq2Xk/xQpDYvNo1mD6lvpKTdNFWW3ixkF5uvYZVGW8JETXwFMkSZ51uYLR8qKZHHDYWwnTxtN3kM57g9GA1S9+G7hZdZHzmdcdxGzkuA5iKVP6KJemHFzcRTJyvkuoduJXn1bUvvtvP26BFVsvuWYv3ZXaVG+/p3jDHNQRbNpeBaTWj6bALkPW+jFq3vAeqz7WUtUGFHtWKX3G5YKfQUPuM3/ewEnSU6SNgZ4YpUUUT26nlp/pW9VYp2nhy/qpt0W2x8GHxr1rKb9u53PKGhtE1t7bT/HoUtpTDgHBySjLxHXoRvDp5tGMuWwrW6LmpeOOMV2dMQehvmUZwfNMs9TGUPRdYbJpoqq7pLzkiD/kPPuDppP4L8wzxjN3qRbv6TS4bhZ3H3MM3vLD1bo5EiWMA7ij0hmBuVmD0UbAGU7EUzVVGdqV37UGvRKyfqrebmoPKHOuoZuGCr1XA85oj9CWLaeLeVLvVXUzGcRIdNq/lJKXZnq2fWbg8/gKrPKuqj8KbKhEvzQhfh6bHptmJUB6j6AtGYRJZrnvxepsJAPqEjVjp/CjwvsLPk7G1R07W3Q5ka24TzuUzWw4ubeJtAqzqydoQrH89Ip8NSbTVQkFCFOOlBwXU9xBfe+yfDP4YQdYuvkGgVFrxZXSIXJL7RGO8X/Omcd8c8RbL+cnDdDeFM/O/cSf5Fn3WzT8+nNrd3cxgqhu8GvPQ0Zx3Xl3j5t0Va6pXXDDOznRFcEB6ocW4kbu2dCNHwtXo1M2/G4hjbjjBDHb0pgfkB/jBtHNBnlVDQd/KtBYUNtRCFKuGeMTTYp1MDOe/PpONb0xpEPNfP+NpYfTxz2EmLL7nTMxaqxJ0SM/tr2VcUlytexkjkZeDUb2dXnWA/K1f8szjVyaeLNKHfPLaUPXFmKrPHvbP18u5sN3+zxm5pUFeE5ZRxLXW+wE2DEPEv/YuoiYPyfvX5x/+kzZmIFqM/xJNlePf8nvrMWbT0yf99u/qOpbnFs9z5+fmjqo9Np5/5XNsPt0Xr3/KVPBgdZ+qLZ/ezEMUaJgvoLzdUJTQWPLX3mcu/M/qcBjjprwcmo3Et8ZpdXo6mg2Bvhk2yDOwGyAWqNfaKv2O+Oyrw5wzvIGcFy3D2Mvyw+ebrRzTv1Wnf3Re7rrvi1CK/7V8fIbNiXd1knSv3HPhcX38a6C/cZ3nbDXrvo3z6/0PTKODtC1QMqc3Nu88nS+jWL5Cf+11ybhSnbZ48mgVrNVMJ5OeDHrGfF1OEgcupAVGjkfXjmUgQ1pzVUTXKGtDZU1JrFTh5qLJKYHW5xR6SC2CAzb0YqC8YYjzckZTXs8qhgo1MlCsewUmUzNTFnqd5YaNFTE7wymV9ZDM6bRXyCjKtgTPJjcXMcs37UjUrcT8i5MDeNmbkXoVkeWYtZJNn63yeP+Vl/4CHdbxVGPr1/yBttCwetYbKMdZZkOxENS6LYvyNTgd7RnMqXNLXH65/mzXYVMGZ3sv8qL4iSBCGRjPrXwLZjKgkeTTOBsRmQUBchvzAOWZZzMZzmMxSyjNWAu8JNF6nNgp4c1h6Y9VcPWhUW/AAj7s1PLBU8fKkfNa4k7QPU/uJdI2BwmmRT6i4uD/Cd8V/m2Mn4UDrB1vMRPz2xR4FBxrQm+WZdAyB4hrHxCkV/R6mVFRDPo8L3ws2/n7BAHnYDU2eGMF2GGHzkdsX7B1/Pr3YcrPY1tE39dG+qyJNf+tMXp4Vx8FBTXnyNaJoT7ThcGQVCEedPpbCPtXsMdHQvNcHmL9O0hU5w6e8cJvsQAXGJ/XN/HWy5M8Q2HpFbXwHRLBWHpaw3fV5cF0eVCuKP+M5Ba3/gHDqfpKSZ5gW6QUCuXjbn5yHtSzWT3sC/ZL6cXWCX2WxTHSP06xYFZHaaYR6hzmtfsq6B3gkwMTE+8qdRXnJrAJOOKPFAUe/0vvRSp8Jbp82nvNXkfvf5kVOWyfHDsDYdy8C9/Gn3nr5nGZ6gzZPmTBUM6qWk8BwAqqsWQCcWDbf6SvF2k3Y4lwXzYWdSZAoAMUtCbWGjWXGXOROEhw+ZuGP9MkRNc//cznYDbHYbVRlmNN9dqk/j3H+3cVCVFXpvuezXfrSnC/1oY2Cn3VwtFW6Sr6I1ufV+zDJYfLbdp8rQ59tvMLaTUqNrgfIaBHQTtZtYtk1Ke8DDDkjvNs+5HC8lmYVP5YFhdBhNtF7jFSv8qRXpmUPjN2KyEvyrlSKLQl1Gk7H3P4IIu5PNBS6J3U2Tdc9rRVPCryvW1ud9u+kv0ybqz3PdzoNO2nc/B9bNpRR+TnoHZTKbOvpGkP6oXY1TS58hOzlMiHS0FYpvB7O3P2Kiw3tAqJNBcBVxYGv8bbQhC0FVnCCOcmP6QT4u/5dZxCANz8uaQqnvbCxLp3p+Iyre+lB2W8cDZLvt/3SOlHl2hWpRqXftSa1u+nMp4XH8zr4BJ/sChQjjpWKaUOqKHtPzB0bfdkJWnwz+dnTMLD/d8/K+Ff4upCCjUdbwdhNJ9xOBrbH3ajuaKSPFHpz1CqN7hM74GMptO4Aym0Or3Pxu/qeV7o1f7i/yN4A4MIHF0DhnDXpLM4RKM5P/bv2AAuGaFXqMZwrNY8t3GwNtVT65YF3ODkHsJ+y35ujOUEGJ/ryP/ViXSgqGf2CM6EH9s0s6/ODq5cxOrwu0+mMm2uJk+M7c4uBHRLg7V5IHcvtjP6x7d2K27fMzR1X7tQlGsw9t3RG5ZhS6sMlv/DEDl97owQ3N8UMBgrPGv5A6uSgVeTBZHZDy5+DGaYUySUcfLA6XlPUsXUM1YbNN4crB4tzEhxZA8VTKoOwnFtyrnkflN2TI5QuA3vwc7cfX1ifeO+rGVI3yX91Wlu5bMvU//h9XLKb/o+bzRGPC9GZHWO75XPqwwEQ7W1oqX9vjhM9fLqZ+qnr+gaLuHD6VGcmP8w8W+WpEf2W2Z3Yx0cRK7t8HzulIXmTAgDc5wmYoNoHd33u4cjwsmZ2+yHnmZ4VzbMndYRGKeCHCh6qfAxnsMy2Hk6RB9QRAlP3dYwfdXV6mHSVKi6ae3BUf2FQGdWqZ2b9Q1rm7aUfLj/RhuBWfa+W9s6Pxa3qwSSBQ7vJ3PE8vSlvqcQ8ABqXxwUYNRKxS0ELWGDutPZvAWRjf3G0QMVm8s32GJXe1mqy8QW5mkiiOfKtwMfo8j3FLc8Pm1UkWPAuDwptNMmAjRxl6y8OUiWO2KnRyFlCFQN2QX5g7Ow9Ipc5QtfY2NdmYVtGWHf5S31GBFUFlJ3RMX8NXSc1jvpMbMhB1y6CxbDSE4Lgxsrdo1N4X5THBGoMy4SHDrF8/zhFVPMBRuPJCDQ1f3ACXLRyZ1wyT5g3pAHntvufHhpVQXdGhyfe4wLwbBZrv4ZR2YZ2hj63JydvNnDubbhmn/KHkbFz3BvZFnt1a1LHkpq3Qyn/1ZE2XN7Om4/rjG+PaMqNGN6erHcOmYPD+OoVf/eqfHkDOcEC+nvAtOv8YEWeC77mrQz0dwXCZLjvmB2EDjGnU0SWqwq2Z6HxpqLnLM+A6ATDnn2n4mp1EOHpOJZDZNY78eRL5BRLbtzXQ8/Q7m1o53ZFj0xnZffKqUCuh7XC36ZuwHy0JDh0L1rLdBdZBnkRGa7dXu7YerEe8+FFzqLRO9lv1WYfAbF1yn77LXpfs1vHh6YEYVb1nOUutqWV1hXXrzYNIOqEbNYbSWSPfXz5j/Dn7zP/tbl+xySBe3CAKTFpLZeqlTQNMTjO4vIAExyk3cVXBLJnmCEUpiMM3S2tUfJPM/oCR49xEeW1oejWV+fOzpVi4qu6zXj17S3R9zkShumWP9RX9ZzE2+J/+SnKWAIWNHcXVdfbg8NALg3Dz7zgmK8jkIGaEo3a/O1/vlTgkHrlGM+oKKgQjKRhwmjj/yrOqw3pBGzfGdchJNqMFSHgyvXHfLtHPd/e5+ff8aCmPTqbT/SVGtDXkjSkyhpVuhPqvLuKT7UcbnRj/TeQr+Fq5FJu+p4BXHbhAnqGGLPu2m8lPcWCWJPcVrd68MBDzJt1GqaRnSKC4ZT8YVRZxtHrbyAzLmLGUjCGnf8w61+Wbn4uVC+BP/KlN2iFt2YqvQMbIKO8WyvKvmsC4JrzB2vfZFjGNBPPmvmLwC3+9JsxMnMvBdWKjX3+KDCzvUtl6T1yvS0O7h/Sd4qsUuz02jv5qcxT3CWbt9z+BXV/Vcpek91B/i+W22yW1y+Lu5YGRn6Fzj9t4yOxCKwt/AqX+CmG6TlntwXfdY+S2GQ/5L8Gpjq8AkpDhr0v5nEMQYCqVfe+z8w/vNOYtlRL/m+Rv9WtuSZW8XM7NIPeTbrGkGv91HPH1+1t2C+2YWVPVOyRmhe1b9Hze+0RyX6tFzq/m5CvnqqOv/jXgNlS4XlumzVZ56uaWBxcvsn7bSLBPlDCAXs+0Kqw7F9vbZVdhE1bnvJXfwuGxi23PBi4CmgPrnmT1YfLrNqc9O+bzxgz9tNdvw9/CVr3KnUSDXwjhT/YhzSlJunznQZYrEZqcQxcJizOcUy4HNmFvaF0j+mNr9yqK5dxCSQMBU1eDzWKZ6xTF0AuHiGTllk6ZzDMi7YF72DhmPY2m8bJBHzdYrnqXf13ugn8i8XXZvOVDr3DV+hkNyvr2dXuoomhX+5KeqcNDm9yyfb1EhrLWbW52HYvmj+2tlyZxjWWHn47e+bmpPTyuHwTmkl2hHp4bJztb5WJWhYeWMGEdUNNsd+bC+XxpV1Q0lzpqVKuJbvoBZMQuvGbh/2JepzfAU+jKqgmqW8dWqYiEvXwiJXxiANlbLE6R+Up1vxOVfd7jVu8dhUsUmAXGbs5h2HyG9a5y/W+iRdMqXvmY0mMeqTm6X8dWVnhDmnS1o7eqbzyY6zKu/6VTaOAbJxKPL2XHXBd/QL0bYHn1/tFrwjETDGFMe3qIsP+cYRtLHtX+fiHSn7q4d3CJwMPyb5aMHynL3QiWX8gb6NtCGyhH2tF08KEKmSt/XNnwPNfgGnZPVQFddYlYaRpBqXfKMG/8PJUqdRJ1tLq0eIxFQm33ZvV4XXnkRmZCX5NJIAuqUtE5oyMJwbSeozkZ9HEfQbpVL7Xfdwq8XXq7EH2qqQSfZ9dadNxRvfB6jv7E0Pbq2i+MFqxx8nXuqSGqIjnLeQUMbALLRbhS1Kjc+3rBq+1i91f9hdyL23xdXnN12sh4QYE+zqT9/K2cR4pAZYJBUvTN9N/5PxOb4VoL5G5jfmOPbyNgE1sSBMdh2FpWLdkZeQVzfuLxOJYgfOjQ2AWrqGq9zRYgxKkbm8NWMx2TN2zzq6cUytfQaOd6fFpM/+hfDqs9NnrTfa1t9rmWdynbBxr9Bhr5WWRMsymqUyWqF71bnnfRwPMvTtYpTZZQ0kGiUtoNlSXxbvq5mnIN2yMTrUIPNY/6Vu67sfmQ2+1BmcUX6nSXjsTNOgNnTmqnVvPlkI2avOHSuHrrUObbUGjEanuKzynk5jbv6pRZPqkjVHP7RxbIgXWG42ZnyNcMVlO4bjGgMtbMntxhPOqN6PPhhCdWhiPh5mIZG64Vr1iYnR/BqPPfmuxcY1T+nlPLQS6zc3B8U7Fdi+HvEL3Pmo6NZy7Wjj/BGgZRqT9b7J7HvFDyvXEB9+eDejDvWdCZ1ZY44qUsBxpDfoSCbH0WRypkdKF1+95F+ZLTEDPcWZOl4Px3rGHnPrMQVYXCMmCmzQw/ZV35r4Y/hK26/z1tVsnM6DFSrQmRsRJbA8MYo6oZ6w4RYJfNsbR/ybvWvlXCvOD7xUhWRh0SUTCn9XP6RvZW6hV2Gsr4wnPA1pfYxVpuCtiVzxYZSptvdrLvHO06rGWlBL2aP+Mq76nHOB+j7VEjcC/PnjGNX9wemcA1r9ZoUpZ64BMmr3+1EJXvvvRPFmJs5TWZVuWg2l8GaWiLpO8/JbuqsTUX9ZjKJP3oQVv6FROUIqovPnRoo8qLhyQEVGT/PCGx1kDD96oBFvnGQKGcwDFDzm0qvWMZ+BDtnbhp9Kn4RvBOuqMy9QNG5/YHqcwxxpiBsM+b0WfqODvT0NkBxYrTwx6N5p60/cuCO0W7ppSH9AYXgTTZ3QuHhdnpm/4VWl/HNZRKlrj2MgPXjg9fS1P2HvbN+T9RgZWPr6bqXPZ7Tt/Wd5r61i5jwuk4XpnmNtT1f36kQjWRDTUaaXP3zPVHTT6Z8c4iSVDaLb2lK+r9PkKYQJ8+bXMAgN4xvzVCfn+7nqkTU/PLhpPzLAZA6xL9OX7CUFSBznHKcH3Tv4XHbeDyHz7qR/TdXJ3Q7EeLLkD/yXvHT3UJG2WXFWJDnexeJQrP8Vmc6IgLK9yHXvPqlVRSCdJfEvJbvzG/s+Jv8Y1cZfa6G2v5t4IfXD8NS1VTgT3j8Wm98M7fVwBf0BMDmyMUbT//K6MQXr1CgAk2VLHiYXMLsX5AH+Sl4e7vdqLmBpU+mW3AZK0RB8ySX1FJ0e8Qn9cWu0V0eTUoASwmZbgMBLLEg7e6JhmbSJtV1TbUDb9P4ADDAafEQoOqSp9B31oUUbZX8SqLJB1+jqLqqYr99uGwDw7LtaJ9uDh14/9KvnYr6qqOoK4Wx1TT5Knd9cqZMdzsY4dkWVnVxn2X8wQm1yMdeQrHyHeUAjn+nASNKJNzfHl3gWN5RxC4qVMWtXMkyfdhBjHgxVWS2KYpAPiS0bn6gDMzkzqybA6tm3mATG4MaNyJ8LmWc0EuGBPTqz4LmYfAYvZFBVAJZt18/3U8zdJwvUBiCEMp8SaEZ2pHqPRJ1ZsaWIRL+HoNdRD78EWGvQn2KbuoTbbVqBRBjK+zkG4ZaHr5yvxfU8mxi/X9XdeUzmp+rv4A+tvABGCtSS+HkB/Wv5VrL2n/QQqYD69gTiIZa4m1UnO4O3xvPs8tmJ2vlzAR4/m/cXHQJ8Fa4XKeQe6Q22wmKFtjCTBV/+mAeNTEcAR/kK2HKnAHcVUdE7HvQOqJVX3ndcrW4ep6Rz1bYEZqd9oOVtmGlxmYT0HUE7UBLZ5FMVaq1Dw+wjbRHwvn9qR8RvfgeZu54YbGhg1kI4QQ3pXIYnRt1Bx8QEmLf7aaNHTXgLqlm7T4MQr4qAOebu/IuY6UB8mwB8y9WSjR1npEHGUpj+8pH3gM/W0UrQF4bjdP9vn2WH8vi7Cz9DfBYU87s9VkTp82M5Lj3Dy41r5SDw+Zs1oVSfpRnjnCE13t5RrEUsRdnXjlmPVGTmkxDWALn4SU7BNaE85oKaaFhvwFNgxTZ9ARJYlDEBNl3Tw/FXnXsCgCPYOMPM6ldzRtkrpEKwQrrLZ+6Kkp2UmsOoY/t1Y7IjVPMt5tp4+6UfFlZWpPkqexzHNrR0+qbCtpNU9qYvXYtrR5iuXrChPfIHMwxfhmrsOHD11zdmBa58xAxTO5VBl0LM9XtVU9OZAiQnCnEi5PZ4Tvs0d67yZHwFNeiw7+qj+xsVkhVS0fgnsZNwdy9HSJPj9zkX8uYrOGldCGmyj0hGx8RgQBNxhYQ+DpC3mGwH/vNJ9pLrw2e5QcCrs4LAtlEGeaRkH+iUR1uYOF7WcvRhNTlwRwsPvhJGSq96lEAiNo+QZLRTPqAx/xbjbzSmYm+pG2l1U7Kc4UzrGB0q0mLADJzqdQ6jaIHWLaVjEz5sSL4HUAHxqEvsAXo2Trw5W/fXMUoRLdycO2jnxKMbXzr6xl0n0RDm7aQWqFFuq/wMuBedoeXbvpwoDoUUWJ7Z4EN58jDoeJcZyYsXhGp3XLa6k+WAV9VObw9HnZlq1s7w93Pr4GTgwe7kqeBE7DWjbH37NJhJ5nvCMSo0j4/RnhR1jnaOOuD7qT3vbrdQ9d3dUoChfOzkbC1tjDzuv51NJmJb3NNVaVDxrVqqv3dtZlsGmCWr1b3zcJnjlvtf4Np6Wbcl5Y6gdmbCWFQoDfTeRFwkjzYwpeWzPCKWHEMSkW3mi4tq/M1tK2ZrHW1KLFVqEWL6WpfUwl7cDHD8kcB4S3m0CG5RLjuUQXfsOJaLlUUobJZdVbszkCXB0AOPkzpNUUccbPn/+SW4MpE7FUK0pvEqqpR2LRe2Q87YtzlSuS5WU2/eWKCnbL9tlgVSVp9lAUCNoI/tXiBdbzcf+xLGt0mRY2VwuJX/Dqk8oABk9TnmG+62AwrUFLGpVoYCWQHe5CA/UoRX/mQdmHO5rIH6JQGwvP/ezg5kbC8Fn1U/7xI2EsrJwnH5pU6ULAc4BvTyqYcqE/hTRrMXyFTpL4QDJoYnlgQBB2XeIvP4Xf3+SbIUnZdGy+rjTKIXdS7LmJkbDXDcrU1ZPYdhUSTwmcG29OPC6TSkxOYFAFVFZ6DgzJGsVptDY2mZ4OvBrC54/Bdw6tx8dtD6JJSUwxWm8/wNnVKCS9ugfWAjJZzq/M5ZsBMXQ+8wDe/QBjCFWxn4VP0kKYuI5FBdYedS5rCe7DJE0GkThrWsSwTXqCdf3Bp3VHrf4mb7VjDMgJxnm0sYZhEoFXMpqrJa3nK9ye6pwL9IlLx61PXMDGgn+NHmzGbCZ+nlRq3l4Ga1o+MRU2S6Ih1IB98ty7m1JvldPB+JMBWaMPLFtDzbSMU6VK78xaHqgq0Lic1J0VjSeOhu6yDsBaGh2K+T38D2yu4XzJCmSZb97MiC/PfhYTSB9WGXOm/vKl5o1u+ofwp+FnNjFrD3pyVnnxiZbh11PTTnn9dB8OASoKqv+eIFeqqpmJZ6U4DdPobuWdb8i5pwuCr8Yfzf9J9pOmCBoNfr8+bwbyTgS4l2ZtjZrAFrPm6qkdzkprDUR/Urw0ZQMHK/tTbSyoUekqcsVyPjHPO1RSfWPFtZ8yQYyuXGmYCJX9sJpQZmnd2iyd+3mZmHb7Q9fUdehy6xSrhKfRhyJGySDYmcefcr5gnEsm8dxbZuZ6wyHsFoMjSIaRIBWlcNavmHfuMeDHTAkmoTT+6Whv6uYqOddoK/8PJUcPl1Dgs9JHRWNb0b9HwXgkSYYzZwfvY8RoWFu8SVZ4Ge7M5IdcVzBnmvlBEtxpjSg0WhRDwM8j0HM2wq8Dk/f8fprwUGJlJ6MO3SpOHFnHzUO8qrfmSbAdSGjn8h3XTPo+e49s4i2Xz5vKPEOOoqgb3BwDSsavYRkOqZlWo1b3WcqkoGG8mqgGNEzYU+BSapAkkeJdhavprO6npaSvVmeFuUCbpKVNZtSS6eKd2hPrbDTUW1oaxEtpzCTE7UrOUbO08O/0FTavS3xcmGx2A7Lc3L6iTe4LRy0DeVqg6GyEYu/gZ9bzQu1jdq7+Bm+hc3Z9uqcN11Xh0HfSIM4a0P9X4ncJhtP/hPyefSLRTZ/VlVzqP/5F8o5y6xFFJz29I/N6kK9HzBQCe36R0N6kdrFNCcNrEtDhw/yvGMdsuh/YvK/k9zEqOsCF0mgT12AZfV9ObFy9V/dqQvVHaEXWGYrpbKQ8VCmNXy1cU+5XHyFIO+eLtYEDGyvxSTCzAwWpXV4YLTy8kPTEnIPP2Mlmpkwq7LLDzn8QhRJk9nQbg9mrVCfh8QRYfoxz7btlPnIWtrdOsHkVkYkzE3bQ1FYSA3EMYqnHBNiJ2VnjDSr0Fnzj8pUXLWNW7sz7l9XLQWFVH62TjqJ6mS7uE3ySdyqgBGFRXrtMLQcRR4jZ2+52bwjzkLSXkLHpH5B0WFtjHd6hXP6i01ny4V+e01Y7GZ/v4PQuhzOp9eXBbZ2CzOpe9KOoQPOAl51J2gvQfMmPlz/2HFcuLjVd3BSN5pbJEBvVy17rmexyW5cZktOHIGmFg7EJdgWXyJdtmc1WWr1KMJyTPsq9Ic2TZmMJvUpmEZYtoZEsBbRfKyfXOx3H/ZWMg4IbUfuVn18QHFTAm0NoGjHVC8G3H1zmyDhkyyj9IKRmcNvDfrf3yBCkHhJjQZnXvYjkpU2eaM9C4OJDZxYKn9dEVb8nVgXuSxmlqQjm0V8WCRmgQnWI3SDAqkmU3cWmtD8f2QdNTS4KBjgZGwyhN2JtZEzSmFC3UmiPyNdUctKSsYjv/JWrE3D7K5W9p36fn9dEvE+T11Zw6F/a4T+htkR8UfC6Ti3bs0CmdqB4o5LjeH4gtjPxe+RphwJWeRDiVr1SYKtNcsXX7i4Pl0LeCT9dmSjqtQ/WUQO3xX8pR1OqpQhxoDNS06lKKG4rq/VKljKazzhmHnjrvZS2Ht1W2T5l+HfgzYYyiLv4WC1u2Mbb4rLyfE7u4/qVkjTvBsRE0pZMqCZqkz7L0y6NHO7PMZyP7QG/uptOR2WGvNSb/MXyW+IJH+BVPh1uCFWVQ0GSsfpXSlX9F/Bd1eC2IBMuX8mAuiAC39Z1vEm9cl59FxcqZUYdiOLtVSvX5NX/gzNovn6R8Le+PH7Ro3K4uNvgYYPGiRwJmxMeHDZzpdtRBxdlIL0j6+v6srp6OJPL6ajv6q5XAyQ9Bg8dkHFd2nzRQ0o5HmTlmQT8jvR7Gz1r8fFeGxKP9Zl5tqXjm/bafAG/mQ/R2tAM5f8GZRRgR1aTVfP8vT+CDmN3I6IBtu9AHJV6uXrl8v2Qw3UDUsGiqQ==

LsJL/zfrrr59zzz03Bx54YJX1K88555x88sknGTJkSG2XuFAzZ87MnXfemTFjxmTevHnZbLPNctBBB6VJkya1Xdp
CnXDCCRk6dGj23HPPhV5s6oorrqilyhbtjTfeyIEHHpjnnnuuyva6+uX4+OOPzw033JDevXsv9AdIXVljscif10n
y3e9+N+PGjcsNN9yQHj16pKKiIs8++2x+8pOfpEOHDku8OvLy8qtf/arK/fnrYg8fPjynnXXZazjjjjFqqbNF69eq
VddddN7/+9a+z0kor5YUXXkiDBg3y4x//OCeffHL23Xff2i5xkkbbddtvUr18/Z5xxxxkLf4+riH32vuOKK1KtXLye
ddFJGjhyZPffcM3Pnzs3nn3+ewYMH16nv+fO9+OKL5d8l//jHP7LCCitkhx12SO/evRdYc7EuKOJruoiviyeeeeCJ
DhgzJa6+9llKplPXXXz8nnnhittlmm9oubQHrrLNOfv0b32SnnXaq8r3z1VdfTc+ePTNp0qTaLrGKpk2b5rnniv
U+pVdu3bNBhtskNNPP32h34h34fq4u8RloPamAtP3dG2bdtS69atS/vtt19pyJAhpRdffLG2S1S1qiXr16lT755JPaLmO
pbLHFFqWzzz67VCr9v/Xzpk6dWtp7771L1157bS1Xt2jvvPNxddff73UUKpVKq666anmXFmmNxbpxi2ir1krKqu3ZeEJtCN2/Z5MpgRVoVj6fXuxZQ5stsgkHJ7wmvpHRjdk3sNsKgl2QXVtpgL6ogacTjp8CNDtjcPsRCd1u1i/Vs5mHe0kYzDm2yBRY7g48kGfYWhJWBV+4NqjpYm6dVUNGnzp+HmVrrMy3C0WbEfnDIYUBHPsT0DIuhrvy5nRnvUHWYm5bl8Pg1lwHbhGN1DY5mS5BRXW5CNa5oaYXmWm6ftlhhjhSoAc9WlWYVLDTE4Snnmz+TEwglXIZRq1fU8t6+JYq4jRi+pwl5eMVj6oHGfWzqJIDSRN
LsJL/zfrrr59zzz03Bx54YJX1K88555x88sknGTJkSG2XuFAzZ87MnXfemTFjxmTevHnZbLPNctBBB6VJkya1Xdp
CnXDCCRk6dGj23HPPhV5s6oorrqilyhbtjTfeyIEHHpjnnnuuyva6+uX4+OOPzw033JDevXsv9AdIXVljscif10n
y3e9+N+PGjcsNN9yQHj16pKKiIs8++2x+8pOfpEOHDku8OvLy8qtf/arK/fnrYg8fPjynnXXZazjjjjFqqbNF69eq
VddddN7/+9a+z0kor5YUXXkiDBg3y4x//OCeffHL23Xff2i5xkkbbddtvUr18/Z5xxxxkLf4+riH32vuOKK1KtXLye
ddFJGjhyZPffcM3Pnzs3nn3+ewYMH16nv+fO9+OKL5d8l//jHP7LCCitkhx12SO/evRdYc7EuKOJruoiviyeeeeCJ
DhgzJa6+9llKplPXXXz8nnnhittlmm9oubQHrrLNOfv0b32SnnXaq8r3z1VdfTc+ePTNp0qTaLrGKpk2b5rnniv
U+pVdu3bNBhtskNNPP32h34fq4u8RloPamAtP3dG2bdtS69atS/vtt19pyJAhpRdffLG2S1S1qiXr16lT755JPaLmO
pbLHFFqWzzz67VCr9v/Xzpk6dWtp7771L1157bS1Xt2jvvPNOldu4cePq9NoxRVtDqFQlT7//PPS5ZdfXtpiy1
Kbdu2LbVq1arKrS6ZPkCfL3k6d6qXPnzuW1IIuiSZMmpXFeeadUKpVKq666anmXFmmNxbpxi2ir1krKqu3ZeEJtCN2/Z5MpgRVoVj6fXuxZQ5stsgkHJ7wmvpHRjdk3sNsKgl2QXVtpgL6ogacTjp8CNDtjcPsRCd1u1i/Vs5mHe0kYzDm2yBRY7g48kGfYWhJWBV+4NqjpYm6dVUNGnzp+HmVrrMy3C0WbEfnDIYUBHPsT0DIuhrvy5nRnvUHWYm5bl8Pg1lwHbhGN1DY5mS5BRXW5CNa5oaYXmWm6ftlhhjhSoAc9WlWYVLDTE4Snnmz+TEwglXIZRq1fU8t6+JYq4jRi+pwl5eMVj6oHGfWzqJIDSRN

bNm1y1FFH1alwc766WNPiDB8+PI888kg52Ey+uEr9NddcUyfXKky+CAqff/75PPjgg+nZs2cqKiry5JNP5uSTT85
PfvKTOjn1/7PPPstvf/vbPPLII9l4440XCAvr2rJTP/jBD5IU54I3hx56aOrXr58HH3ywEO9zJ598cjp37pxHHnk
ka621Vp555pl8/PHHGTBgQH7xi1/UdnkL9dxzz2WPPfbIjBkzMn369LRu3ToffffRRmjZtmjZt2gg3v6VMS1/Gnn7
66To9Cmv69On52c9+lrvuuisff/zxAvvr2odFUsz1K1u1apWnn3466623Xq666qr84Q9/yBNPPPJGHH344xxxzTP7
zn//UdokL1bt378V+INe1aemrr756zjzzzzAVCzGuuuSYXX3xx3n///VqqbNHuv//+hW6vbhwI48aNN6h9-666RZ587
LuapFu+qqq3LWWWelX79++d3vfpfDDjssb731VkaPHp3jjz8+F198cWW2XuIBmzZpl7NixWWuttWq7lGo7+uij88g
j/1979x0V1fWuD/yZAVRAmh0jIkVRFBQkaMSGBVEUW2KNDTVqbCH2fGNPjDFGo4lGE8WuiC3W2ABRUIMMgggWWgWg
iakAsgIZyfn/4Y67DAGKC7DPwfNZiLeecufc+4Q5zznn33u8+iZ9++glubm4AgNDQUEyaNAmdO3fGzzz//LDjhK2/
qAfk6ufWDBLRvoK2bVqojddrQHs2AWzZsiVq1KiBTZs2wczMDADw+PFjB+8PPs8c+cEJ9RkZGSEM2fOoFm
zZqKjvLVKlSohLi4O9erVUzohLi4IRo0aITMzU0ywImbjgEhhn4+oqQCi0a9dObfd6uTA0NMSxY8fQunVrteNnzpy
Bp6cn0tPTBSrXFFtpwD5rdzShYyhoaGiGiIyM1AITMzU0ywImjbgEhhn4+oqQCi0a9dObfd6uTA0NMSxY8fQunVrteNnzpy
Bp6cn0tPTBSUrXFFtpwD5rdzSthYyhoaGiIyM1Jrex9WqVUNQUBAcHR1hYmKC8PBw2NnZISgoCFOmTEFUVJToiBr
at2+PBg0a4Oeff4apqSmio6Ohp6eHjz//HO8++49JLKCYnKtU8//VRq1KiRtGvXLklfX1/y8/OTFi5cNWpU0f
aunWr6HhlhqGhoZSQkT16NFDbWrx4scSQbzgZ0LkkiT16NFDWrx4scSQbzgZ0LkkiT16NFDWrx4scSQbzgZ0LkkiT16NFDW

KNm3aCEpVuNdneri7uyMuLg4RERGwsbFB06ZNBSYrnqysLEREROCPP/5AYmIiatasKTpSgU6dOqWxQyUAvHjxAmf
OnBGQqHhMTEzUXiuVStjZ2WHBggXw8PAQlKp4tOXBY/369ejZsyfq1KkDCwsLKBQKJCUlwdraGvv37wcAPH/+HLN
nzxac9BWlUgmlUql63a9fP/Tr109gouLRtoel06dPo1WrVhg9erTa8ezsbJw+fRpt27YVlKxwlSpVKnAn2/j4eFX
hXm60abZCVlYWPvnkE8yePVs1q7RFixZo0aKF4GSFs7e3l82s87dx5swZnD17VlXYBAA7Ozv8+OOPcHNzE5hMnTY
ULPMbP348Vq1apXE8PT0dXl5eOHXqVOmHegubN2/Gd999hxs3bgAAGjRogGnTpmHIkCGCkxXs9eLlhx9+CAsLC1k
PiMyfPx8LFiyAi4uLVs32zvP8+XONe4yCVueIEhwcDEmS0KFDB+zZswdVqlRRnatQoQIsLS1Ru3ZtgQkLVr16dVy
6dEmjuHnp0iVZzjTdsGEDKleurLFic9euXcjIyMCwYcMEJdOkq6sLS0tLobP7iUoKi5sl4MCBA6p/e3t7Y8aMGYi
MjETLli0BAOfPn8euXbswf/58URELlZSUhJo1a6qWptStWxd169ZFbm4ukpKSULduXcEJCxYcHIzt27djz549yMn
JQZ8+fXDw4EHZzbKJiYlR/fvatWtITk5Wvc7JycHRo0fx3nvviYhWLNo4rV/bHjzs7OwQGxuLY8eO4fr165AkCQ0
bNkTnzp1VRUQ5LX96/TP9OoVCgUqVKqFu3bqyW+qmjQ9L7u7uuH//vsZN+5MnT+Du7i7Lm9CePXtiwYIFCAgIAAB
VoX7mzJno27ev4HTF9/rDnpzo6elh3759shnoKMzrBe5vv/0W06/dPx6JFi+Dg4AA9PT2198rpofevf91devWRVZWlsb
x7Oxs4dfsmJgYNGnSBEqlstDv4zyOjo6llKr4jh8/ji+//BJfffWV6lh6ejo8PT0FpiqeZcuWYfbs2ZgwYQLc3Nw
gSRLCwsIwduxY/P3337JsDRETE6P2OXh9QOS3336T1f0FAKxZswYbN26U7T1bQRISEjBhwgScOnVKbQahHFpC5Jc
3IJKQkAALCwu1wWo5Gz16ND755BPcvn0brVq1gkKhQGhoKL799YltMmTJFdDwNixcvxpo1azSO16hRA5988omips
A8OWXX2LWrFnYunWrbO+B8nvw4AGmTp2qmriQPwPlXwMSFuHWkJO6FQW4XOOBV9kaNGuHAgQOwsbFRHX/w4AF
q164tu7zAq/5BKSkp6NKlCwYPHowePXrIsv8joL78o6A/NX19ffz444/w8fEp7WhlUmEPHqtWrcJXX30lywcPbfO
mJU16enro378/1q5dK5u/S3NzcyxZskSrHpaUSiUePHigMePx+vXrcHFxKXCGpGhPnz5Ft27dcPXqVTx79gy1a9d
GcnIyPvjgAxw5ckQ2/YXfpjfs3r1732GStzdixAg4ODjg888/Fx2lUPm/I/Ie8F8nx4f+1+3fvx+LFi3CqlWr0Lx
5cygUCkRERGDixImYMWOG0IKQUqlEcnIyatSoofpdF3R/Idffb0JCAlq3bo2pU6fC19cXz549Q5cuXaCrq4vff/9
dNt8TBbGyssL8+fMxdOhQteObNm3CvHnzkzJCQIChZ4czNzREWFqbRQ3jPnj2q9jdyUrVqVYSHh6s9k8hdq1atAAC
TJ09GzZo1Nb7v5DzDOiMjA0lJSRqry+Q2MCJJEn744Qd8//33+OuvvwAAtWvXxrRp0zBp0iTZDVpXqlQJcXFxGvt
AJCYmolGjRsjMzBQTrBBOTk64efMmsrKyYGlpqfE9fPHiRUHJCte1a1ckJSVhwoQJBJU5c6Nmzp6BkJBJnbpYAbdy
s4nWNGjWCq6srAgIC0LFjR9Vxuda958yZg48++khtYx65SkhIgCRJsLa2Rnh4uFqhokKFCqhRowZ0dHQEJtT0Nss
xU1NT33Gat/Pjjz/i559/Vnvw6NmzJxo3box58+bJtriZnp6OkJCQAm8w5bLxUZ59+/ZhxowZmDZtGlxdXSFJEi5
cuIDvv/8ec+fORXZ2NmbOnIkvv/wSS5cuFR0XAPDPDPP/+oHj7kLq/wplAoMHz4cLiLMHz4cLVSoDY6A5uEkvIiUGEWYWrHJfjKjiZuxpy4TFCQ0M
RHByMyMhI5ObmwtnZGZ06dRIdTU3+dhvaxNbWFgsXLsTZs2fRvHlzjQcQ2OXxfaOumCvmvfenp6WjRooWqv3B2djZ
0dXXh4+MjtLiZkJCgupeQYzHtTaysrHDs2DG0b98eSqUS/v7+qFixIg4fPizrwiYA3L9/v8DEv31atWuH+/fsCEr3
ZuHHj0LFjR5w9exbm5uYAgJ07d8LHxcbN24UG64Ao0aNwvbt22U/Q/11MTExiIyMVGtjIXePHj3CiBEj8Pvvvvxd
4Xm4DIwqFAr6+vqoBEQBqGxfKTY0aNRATE6NR3IyOjkbVqlXFhCqC3GZwF0doaCjOnDmDZs2aiY5CMsLiZjmnUCi
wevVqbNu2DV5eXliyZInq4Uhuo2B5PvnkE9ERii1vB01tKoBr866m2vjgERUVhW7duiEjIwPp6emoUqUK/v77bxg
YGKBGjRqyKFa87uuvv8aKFSvUehU6OjqiTp06mD17NsLDw2FoaIgpU6bIpripTQ9LeYU3SZJgZGSktrlGhQoV0LJ
lS40+nHKQm5uLjRs3Yu/evUhMTIRCoYCVlRVq1apV4Mw9kbSx3UaedevWwdTUFJGRkYiMjFQ7p1AoZPF9IedZSkX
Rlmvf6ztzy3mX7qI0adIEhw4dQqdOndCiRQscOnRIKzYSSrW1RUBAAL744gu14zt37tTotS8Xc+bMQUpKCjp16oQ
zZ87g6NGjGDVqFLZs2SLLdiEvXrzAL7/8gpMnT8LR0VGjlcWyZcsEJSvc+++/j7t372pVcfOzzz7D48ePcf78ebi
7u2Pfvn148OABvvrqK3z//fei4xVJzkXNPAMGDMCkSZNgZGSk6pEEhKCyZMnY8CAAYLTadLGDWItLCxkOxGLxOG
y9HdAm2Zhvb686ffff8fAgQPx4YcfYs6cObCysplLdyF2eCxcuYNeuXQX+juWyjPDAgQPo2rUr9PT01PqyFkSOTd2
1UZMmTTBo0CCNB4+vvvoKO3fuxOXLlwUlK1z79u3RoEED/PzzzzA1NUV0dDT09PTw8ccfY/LkyW+1hLY06OvrIyo
qSmP347i4ODg5OSEzMxOJiYmwt7dHRkaGoJTqJk+ejM2bN8PR0VFrHpbmz5+PqVOnyn4mE/CqENujRw8cOXIETZs
2RcOGDSFJEmJjY3H58mV4e3vjt99+Ex2TBNKWpY/a6tq1awX+fuVyb+Hk5FTgAMedO3dQo0YNtcKmJ/5tmzw/
69++PTp06wc3NTdX37+TJk9i1axd69+4tOmKhhgwZgj/++AN//vkntm/fLtslm+7u7kWel+PM8Fu3bmHs2LH4+OO
P0aRJE417DDl+z5mbm2P//v1wdXWFsbExIiIi0KBBAxw4cABLlixBaGio6IiFfm8URG7fG//88w+GDBmCmCXbt2qWb
/5+bmYujQoVizZg0qVKggOKH2O378OL7//nusXbtWY4YslV8sbpawN83Cun37tuiIal4vbgKvbpC9vb1hYGCAq1e
vyrK46e/vj6FDh8LDwwMnTpyAh4cHbty4geTkZPTu3Vs2s3Ly98UqjFz7YuWXmZmpscGC3DaDKOzBIzAwEAbJ
88DA1NcUff/wBOzs7mJqa4ty5c2jUqBH++OMPDBs2DHFxcaIjqnFyckLTpk3xy+/qG7OsrKyMpERHRyMmMqKgp
hYWH4+OOPZbNksqiHJYVCgaCgoFJMU/Zs2LABkydPxv79zV+10FBQejVqxd++uknjT51cmBlZVXkw5PcrtnaRtu
WPhZEzte+27dvo3fv3rh8+bJa7828z7Rcfr9vs6GmHGcQLV26FFOnTgUAREZGYvny5YiNjYUkSbC3t8cnn3yC6dO
n4/z584KTvlLQgHpWVhZ8fX3h4eGhVvSWSwFcm50/fx6DBg1CYmKi6lje36Nc7/GNjY1Vy6br1auHbdu2wc3NDQk
JCWjcuLEsBqe1/XsDeNUnPTo6Gvr6+nBwcJDtbPs39dOX42fYzMwMGRkoGBgcaggtxap1Hp4LL0Eubr64s
ePXqoZmGdP39ebRaW3LRr105t9Mje3h7h4eHo3bu3bKd6L1q0CMuXL8f48ZeNhZGSEFStWwMrKCmPGjFH1E5KD15e
ia9Oy9Nelp6djxowZCAgIQEpKisZ5uV3s+vbtiz/++APLly/Hb7/9pnrwCA8Ph5OTk+h4dLT01PdUNSsWRNJSUl
o1KgRTExMkJSUJDidplWrVsHb2xt16tSBo6MjFAoFYmJikJOTg0OHDgF49cD96aefCk76P+Q406M4du/ejXnz5mDs2LE4evQo/v77b9SpUwfr1q3DkSNH8A5uuqfWs7e3l9V1/6Vr165oIMDtz59GjVqhJSUFDg7O6Omm276Q/mEC8M5tEuFdQF4MjeYNGD8hTrhTyEQQT8gXyyEJk/AVsrYFPPTUwjeQBDJ8LHUB6GfTcRG2lBcfJSzFfaJd8vKrD2oqrC6vj4ocSd0DoCddSQy+HBa9yZhudJCa3yqFKlsV6y1SXb/ZB8vrn60yq0OgeT3KqmZP0dWBTCNmnTdvxr0Vf7a
nZMlplsKOHTvwxRdfFFhE7tChA2bOnIlt27bJsrj52WjBsrj52Wefqb3OyhfqVFQUjh49imPTpokJVYTCCNhJSKBSoVKkSbG1tC1tPOvuEtBzGeoA+MqaWlSSJy
t0bNnT9nsdqqtSx+15do3efJkWFlZ4e3ikpRe3ikpKbJqCwLIt/BQXLNnz0bVqlXFhCqC3GZwF0do
bvBXVP8/Pzw9+fn4A5Dm4HhgYQNb//3U//fQTJkyUMqJ3szHxwdOTk7R3i4+NRr149NGGdRsKyZGdnR3i4+NRr

7bc2aNbJ5ltL27w0AaNCgARo0aCA6xhvt27dP7XXevdCmTZveqshcmpYvX64Vf2tUujhzs4Rp2ywsbWRoaIirV6+
iXr16qFatGoKDg+Hg4IDY2Fh06NBBdr0Vs7Ky4OHhgbVr12rFBe5148ePR3BwMBYsWIChQ4di1apV+PPPP7F27Vo
sXrwYgwcPFh1R63l4eGD48OEYNGgQxo4di6ioKEyaNAlbtmzB48eP8ccff4iOqOH58+fYunUrrl+/DkmS0LBhQww
aNEj2fZBu3ryJW7duoW3bttDX15ddL8jXrVy5Ev/73/8wbNgw/PrrrxgxYgRu3bqFCxcuYPz48fj6669FR1SpVas
Wjh49WmhT96ioKHTt2hXJycmlG+w/WLVqFSIiImSzEiCPu7s7s7Ll68iJycHNjZ2UGSJNy4cQM6Ojpo2LAh4uPjVTP
W7e3tRcfViqWPBdGWa1+1atUQFBQER0dHmJiYIDw8HHZ2dggKCsKUKVMQFRUlOmKhIiMjERsbC4VCAXt7e9kOQAK
vBpqGDBmCHTt2qBUO09PT4eHhgZSUFJw6dQq1atUSF7KMMDU1+rHf/hhx8wZ84cWRWR8xgaGiI6Olq
2fVcLsm3bNmRlZWH48OGIiopCly5dkJKSggoVKmDjxo3o37+/6Ihap7DBx4LIsR1SQbZv346346dO3di//79oqMQFt
nbpYwbZiF9fTpU9WSqjfdJMhl6dXrqlSpotop77333tW9XrlyRbRbRGlKAcPHsTmzZv
Rvn17+Pj4oE2bNrC1tYWlpSW2bdsmmmwe81+Xm5Am7GBBk3dzdSSJr9+ffdaxFkReUhQUiTqDpayiRCk
ehqWvfVpI0v/Bn2FNNeRRJhTRYrD+i6evsXV9EgbaycPbW5xmn3iyFMgAk1jjNptEJgoOhxEbzm+4gZgPXzUTo
GDS3sdkomVqhjbSsxwn4ZwdeGtVSx+MPpyvKCZvRt/yXNLh/GFlrUBvHNghmxyZ+Isgn5YKkIKP2I
m/Tjqi2Dc+ZzRfn3vlhHJu7qu96I8+K6mUfN1Xc6kzs3FnJ8fkiOPUW4m9Q3MJlXmBfJ3Ib/Sq5bGJmfcqZgyu+L
8T2FN7b4EBB0t+6dNWyqwIvVXU1Ox0NAxxq2t8OhvltsXpJ7/mUvd2IF0bD/PcQ7/W7L6Q6p9FQc1yMcXrtz0Fg
A6BFPF6MBpm2f69GtiBcVJi9eCgtW4h8TycGfMnJTWjYJMF5t85aj3/vd44FJS4o7PXFsvzMtahBhMdlTKXr5wyK
8sHUnZ7yHfc0rI/xUH9HZKcRnfmDpMY5i4cwwt9CWyrafTK4vvKdqthCAjTlYA9w5xgaXgHyzZ8HpGYfLeaNFPOdP
v+7J4DI9zrgOqAo1kddZU/c6SRBuwkZ3rzA3p3nffsqbFbJNnFfXLAm+Z5NoP0cVTkEv27tXlJgmIcBbZ9HuJg4b5y
3ejb5nYjmhp7l6hkN6D8LjtoaeGa0MCWJoFQ5yyV0cJdP9O6nZP7/ytjBxsbm5YqD5BUJJ2YvF4zbajhdV9Ni5XOQy
Wx0xnfXzVW5iNDV5+MQaXV0NWbDXwvzd1XdfHhz3+7/Wv5bzR8XAlpsiSfPNVCPDdl22qxnrTIzrsKrHo2CnHp+8
pl5vB/9GBmYJ6XY1hg6jhjyEEgeSNJ6JbmPyXdhNsCkPuyCFjl0e0hptQ7Dc5wUn5Drh7Kgu5znBwrqOEOv+/lj
4W3tjVTGp6NHOCKN6c+7YtQL3GCIyWhVGEkUFSyJe1Ln8nr2e6v6K9pSRmHkQ6Ko4GENyJb8A5bCcjvLdO3Cmn+
lbRYaKW9SRy9Lp7k6PXyLQnWkSXh7oQnXIxkBgv1RFbPxB1d5rP7TH6O7tiZ6LjESjEIUeu8EKyUZhvL8lXo3Wn5
GsNMfTpDfAkwgw4hOKmH8Ti0PxWFV28HsMdQxcHwKW7G4TjNPD/QRVqkxyOpv1tzvKtQcHDNKZMJRIqbTMA1pDx
nwFWLM1NMnhb7eDBr6zGh+DB1w6ZpNDeKKVdyeQP0uhPrXJj3qfaT1qz8HbBBTW+ANsiuATEvvgJzcvAMOQYzA
oRbYHmGEWN9EpIrVbAaoN2BjqiYNGvvPmBg+IE3DKPEN9Du37RGEmYs8/Um

7bc2aNbJ5ltL27w0AaNCgARo0aCA6xhvt27dP7XXevdCmTZveqshcmpYvX64Vf2tUujhzs4Rp2ywsbWRoaIirV6+
iXr16qFatGoKDg+Hg4IDY2Fh06NBBdr0Vs7Ky4OHhgbVr12rFBe5148ePR3BwMBYsWIChQ4di1apV+PPPP7F27Vo
sXrwYgwcPFh1R63l4eGD48OEYNGgQxo4di6ioKEyaNAlbtmzB48eP8ccff4iOqOH58+fYunUrrl+/DkmS0LBhQww
aNEj2fZBu3ryJW7duoW3bttDX15ddL8jXrVy5Ev/73/8wbNgw/PrrrxgxYgRu3bqFCxcuYPz48fj6669FR1SpVas
Wjh49WmhT96ioKHTt2hXJycmlG+w/WLVqFSIiImSzEiCPu7s7s7Ll68iJycHNjZ2UGSJNy4cQM6Ojpo2LAh4uPjVTP
W7e3tRcfViqWPBdGWa1+1atUQFBQER0dHmJiYIDw8HHZ2dggKCsKUKVMQFRUlOmKhIiMjERsbC4VCAXt7e9kOQAK
vBpqGDBmCHTt2qBUO09PT4eHhgZSUFJw6dQq1atUSF7KMMDU1xYkTJ/D++++rHf/hhx8wZ84cWRWR8xgaGiI6Olq
2fVcLsm3bNmRlZWH48OGIiopCly5dkJKSggoVKmDjxo3o37+/6Ihap7DBx4LIsR1SQbZv346dO3di//79oqMQFt
nbpYwbZiF9fTpU9WSqjfdJMhl6dXrqlSpotop77333sOVK1fg4OCAtLQ2WSyjyE9PTw9XrlyRbRbRGlKAcPHsTmzZv
Rvn17+Pj4oE2bNrC1tYWlpSW2bdsmmmwe81+Xm5uLmzZt4+PChxozZ3vKbecrJo0SLV53nhwoUYNmwyYxo0bB1tbW9k
VVvJUrlwZY8eOFR2j2FJSUtCvXz8QHxjaafPkbb0fxeBwe7+vd5hUPtIGPMb2hgJQ4EHuWw3dvFBB0ulg2HWm22ke8dNqtGb4e36WrJSa76RlvTb8mqXb
fHDt2THDaV8WfvLY3VapUwaNHj9CgQQM4OjbIavZxzftpy7cvJyUHlypUBvCp0/vXXX7Czs4s4+MFpyvYw4c
PMWDAAJw9cxaWlpbo1xdbIy6lP6VVSy12mupXnUdv6tvSr1hg9ejQGDRq3Vz3DRDwRtyl5z6WrrcJaPxx6jf6VLb
B6xHXfAf7A7Lt9Lzt0GDRmGNuIYKqSgE+AR4GxBJN4KJOgS9pDuOc8qXH9NmP8mGaAxnLyn0BsCqPhCqPVjWzLfkfS
TMQ7Ooro/fyjSTOMkw/cWAQaM/Xx8yN/X7xvhIYNjEJLhDI/4ThG/JiHdbVKVk4VmZLYLpJq7nX6ECz9J1XZHRwv
68/3t8rSaa6cOsXeEvSQYU7pKDCKy+JQpyIZJmqfBHc0dZ9hJwKJCYl77Ggd+GccfD1q9WrWqUxn4KaWqIlTrhmDQ
cVHeyZcUvUjtXv1ZUF8bKUBPD8PvPXtBBGMiOCoH4CGxK4+DRIQzEvtOeUFQSZ0u/HNWVGQvIEZ+KgOmWuuQzcmnQmM
UbJvOmTXtdgsdFhQQmbA2twQdLXTCUmMnuoDiVSCg5jBYrj0eh8QbAIiUnOoKmeIMEDuQVqxZqPBEjyNgU5O7TWh
lcTCcb+QpfuIEnVKDbLdZdHEYiBN3AUnDxqqHXZ5q1N4DkkHYwQBjUuOL8M5QMXDmBjyQfNG7sQSFGONqc1O1HiNv
sMj9Oo3I1wvS9RvX5EoGWvOhhiFC4Q4bQv2zs+AeFkvQJYG9CjSy5fc5Bc5HQcL7bAhVBpvfr1J2SsfR0rmNZbEQG
72bebwG1mJwcufbhCM+chX+6BeFuIvEb6R3n3sSTuJ0d3hUIRgDHM8AD+lOiG8TQV+rO0a3Rwv5vGvcl3oU5sHp92u
s3OIGNL+0YXUYcWQKDkv+u7SU3uIIFoHEG3YgRWM/a8ZyL7ZhxL1I6j4zkHZNvYqHPYAMPzbuAG1NV6mEj8sOf8QHF
H8N2WBQwO8+xGeVxF6Zf+Z9NiNHjMiVLHNHFuGhQhK9Bhm+VmQUhGJ+HLBx8/f3X2bUmXmKQt9/HfXJt8T1G2y8yRu

ZWVlISEiArq4ubGxsZHWPX1ZmIcudoaEhrl69inr16qFatWoIDg6Gg4MDYmNj0aFDB1UvarkICQmBt7c3TExMVBu
dRkZGIi0tDQcOHJBd8TjPvXv3cODAgQKX/st1Pwt6t1jcLGGjRo2Cubk5Fi5cqHZ87ty5uHv3Lvz8/AQlU6erq4t
JkyZh3Lhxajfv2lDcVCqVGsfyiltyLWANGjQIJ06cwIcffoiaNWtqzPSdO3euoGRFs7KyUhWyzMzMIEmSqpBVuXJ
lPHz4ENbW1ggODhbWq+dtNoSRYyuD48eP4/vvv1ftIix327dvx+rVq7F+/XrVDKz4+HiMHj0aY8aMweDBgwUnLBt
CQkLQs2dPGBsba9WNpraRJAkfffQRHj58iOPHj+PcuXPw9vbGV199Jcu+iu7u7kWuFJFr/+Y8N2/exFdffYVt27Y
hNzdXltdrAJgyZQr09PSwePFi0VH+s7S0NPj4+MhiJhaVPisrK8yfPx9Dhw5VO75p0ybMmzcPCQkJgpIV7MMPP4S
Liwtmzpypdvy7775DeHg4du3aJSjZ23n69CmGDx+O3r17c4ZsCUlLS0N4eHiBk0Tyf75Fs7CwwJEjR+Dg4ICmTZt
i5syZGDhwIM6dOwdPT088efJEdEQ1TZo0QatWrfDzzz+rJqzk5OTg008/RVhYGK5cuSI4oabAwEB4e3vDysoK8fH
xaNKkCRITEyFJEpydnWV/P0TvBoubJczExAQRERREao/03btyAi4uLbL7Mzp07Bz8/PwQEBKBhw4YYMmQI+vfvj9q
1a8u+uPmmYpYcC1gmJiY4cuQI3NzcREd5Kzt27MAvv/yCdevWqZbu3rx5E2PGjMEnn3wCNzc3DBgwALVq1ZJF38K
UlBTV8tK7d+/i119/RWZmJry9vdGmTRvB6QpmZmaGjIwMZGdnw8DAQGMpfWpqqqBkBbOxscHu3bs1ZipERkbiww8
/lN2DUh5tuikGtPNGU1tlZWXBy8sL6enpiImJwTfffIMJEyaIjlWg/LOssrKycOnSJVy5cgXDhg3DihUrBCUr2PP
nzxEaGopTp04hJCQEly5dQqNGjdC+fXu0a9dOtj2FJ06ciM2bN8PW1hYuLi4aK0Q4I+S/CQwMxPLly1Wz0hs2bIj
PPvuMszffgUqVKuHKlSsaGy3euHEDDg4OePHihaBkBatevTqCgoLg4OCgdvzy5cvo1KkTHjx4ICjZ27ty5Qq6d++
OxMRE0VG03sGDBzF48GCkp6fDyMhIbZBPoVDI5l7Zx8cHK1aswJgxY+Di4oLPP/8cf3h6enJ9LT06FUKpGeno7du3fLsk9r/h0K8xs2bFgpJSk
eAwMDnDp1Sm33UgAIDw9H+/btkZGRIShZ4bTlpvh12nijqS1e79Ob59mzZxg4cCC8vLzUboi15Tt73rx5eP78OZY
uXSo6iho9PT1UqVIFQ4YMgbu7O1q3bg0TExPRsd7I3d290HMKhYIzQv6Dn376Cb6+vvjwww/rx5eP78OZY
AgrZq0qQJBg0ahC+++ELt+FdffQV/f3/ZDZQVdu2Li4uDk5OTVl37QkND0aNHDzx+/Fh0FK3XoEEDdOvWDYsWLYK
BgYHoOIXK2wRJV1cXL168QO3atZGbm4ulS5eqNoWcPXu27DbBcXNzw7Rp0zSek3777Td8++23Od8+P8yZM0djmYWcxMfHY/3
69diyZQvS0tLQuXNnHDhwQHSsQl27dq3AHhve3t6CEhXu999/x8qVK7FmzRpZziwtjIGBAU6fPq1aFpvnwoULaNe
uHTIyMpCYmIgmTZrg+fPnglK+2nFcV1cXM2bMwNatW3Ho0CF4eHhg3bp1AF7NwomMjMT58+eFZSwrevTogaSkJKx
fvx7NmzeHQqFAREQERo8eDQsLC1l+Z2jLTfHrtPFGU1sU1Kf39ddyb3NSkJs3b8LV1VV2hfpevXohNDQUOjo6aN+
+veqnUaNGoqORIO+99x5mzZqlUcRctWoVvv76a/z111+CkpVNe/bsQf/+/dGpUye/exK5du9C
7d2/REdW8//776NGjh8YmPPPmzcPBgwcRGRkpKFnh8venlyQJ9+/fx5YtW9C2bVvs2LFDULKyw9DQEJcvX4a1tbX
oKEVSKpVITk5GjRo1REd5Kzt37sT06dxceJEtfrFqlWrsHjxYrVrtlwGfWvVqoWgoCDY29ujcePG+Oabb+Oabb+Dt7Y3
o6Gi4ubkJfS4lcVjcfAcCAgKwYsUKxMbGAgAaNWqEyZMno1+/foKTFU9OTg4OHjwIPz8/WRYqbt++jd69e+Py5cs
aD6QAZPkw+ujRI/Tr1w+nT5/WiqXHeby8vJCcnIx169apliFHRUVh9OjRqFWrFg4dOoSDOoSDBw/iy++Py5cs
tGoKCguDo6Ijnz5/D2NgY4eHhqqJsXFwcWrZsKZudj58+fQpjY2PVv4tiYGAAXV3d0ohVLI8ePcKwYcNw9OhR1ec
4OzsbXbp0wcaNG2V5Q6ctN8WvzyiMjY0t8kazf//+omJqPW3v01uQLVu2YMaMGbItDMXExCAkJAQhISE2rdvD39/f9HRinTz5k3cunULbdu2hb6+vqroTf+ekZERoqKiClwm7eTkxAfSErJ06VJMnToWkuZ2MXltAPI2cPr
kk08wffp02Q36HjhwAH379sWgQYPHjwH379sWgQYPQoUMHAK/aGOzYsQO7du2S5Qqc/JsuKpVKVK9eHR06dMCsWbO
GDJD9s7RSqcSDBw9QvXp9vXp10VHeSkH7WbxOjoO+vXr1gpeXF0aPHo3p06dj3759GD58PbU3QszMzOcPHlSdEQSgMV
N0jo9evSAjo4Ofv31V1hbWyM8PBwpvEip06dj3759GD58OPbu3QqszMzOcPHlSdEQSgMVWQvT19bF69eoCd8d+9uwZunTpgrS0NFy7dk1AuqIdPnwYixYtwLnVM69PsHn06BEWLFi
AESNGwMHBQWOSiFxW7SmVSpiYmLxxEExuk1q0cdD39u3beP78ORwdHZGRkYGpU6eqlv4vX75cNjmpdLG4+Q6kpaV
h9+7duH37NqZOnYoqVarg+7uwZunTpgrS0NFy7dk1AuqIdPnwYixYtwLnVM69PsHn06BEWLFiAESNGwMHBQWOSiFxW7SmVSpiYmLxxEExuk1q0cdD39u3beP78ORwdHZGRkYGpU6eqlv4vX75cNjmpdLG4+Q6kpaV
Nm4qO8q/ExcXh+vXrkCQJDRs21OiFJFr+kVIjIyPExMSoRtPlVtwMCQmBm5sbdHV1ERISUuR7X758id9++w1w1BQUG
Ii4srpYTaTxtvirXx5rKs0JY2J/kLFa/PEBI5wFSY5cuX49SpUzhz5gyePxGZs2aoV27dmjfmjfvj3atm2rmsEuN0O
HDsXDhw+xbt06NGrUSDVYdvz4cfj6+uLq1auq1auiI2qtr776CkuXLoWbm5taz82wsDBMmTJF7TMxadIkUTG13u7duzF
kyBDs2LFDbbZjeno6PDw8kJKSglOnTqFWrVriQmo5Hx+fYr3Pz8/vHScpm940mzCPnAZ8lUolfvjhhzf2lpbrpBY
ibcfiZgmLiYlBp06dYGJigsTERMTHx8Pa2hxEEXuk1q0cdD39u3beP78ORwdHZGRkYGpU6eqlv4vX75cNjmpdLG4+Q6kpaV
h9+7duH37NqZOnYoqVarg+7uwZunTpgrS0NFy7dk1AuqIdPnwYixYtwLnVM69PsHn06BEWLFiAESNGwMHBQWOSiFxW7SmVSpiYmLxxEExuk1q0cdD39u3beP78ORwdHZGRkYGpU6eqlv4vX75cNjmpdLG4+Q6kpaV
cHBxkuaGJs7MzVq9erVpeSiVLqVSia9euqFixIoBXG8h06NBBtcPty5cvcfToudnc/Lythw8folu3boiiIiBAdBTk
5Odi4cSMCAwML3HlcLhttaONNMZU+bWpzkpOTg9DQUDg4OKBKlSqi4xSLi4uLqs+mnIuZ+dWqVQvHjh1D06ZN2rrU3Du7u7rh16xY
CJCQkwMHBgUun/4P8S3gLo1AocPv27Xecpmxbt24dJk2ahmOHD8Pd3R3Pnz+Hp6enqnZ+Hp6cnHj58FOnTqF27dqiI2o1pVI
JS0tLODk5FbniYt++faWYikTS1p6bebRloPd1ERERiI2NhUKhQKNGjdC8eXPRyg+TRyKyM+//xzDB8+HEuWLFH
rsdK1a1cMGjRIYLKyo0mTJoiJiYG1tbGiBJUuWoEKFCvjll19Nv8eLFmDJlCr2++usCZ47J9YFPWwpZ+Ud
AP/74Y433DB06tLTi/GuZmZnIyspSO2ZsbIwaNWrIAJAJMnT8bGjRvh5eWFJk2ayLb/XP7Pqjb6888/ERYWVuD

fHmc0lYzJkyfDysoKJ0+eLLDNiZzo6OigS5cuiI2N1Zriply+t95Wenp6gZuP/f3336pBNPp3EhISREcoN0aNGoX
U1FT06tUL+/fvx+zZs5GcnIyQkBDZFjbf1KpHTgNOY8eOhb+/P27fvg0fHx98/PHHWvPdrC3++OMPpKamomvXrqp
jmzdvxty5c5Geno5evXrhxx9/lM33slzvid9EmwZ689y7dw8DBw5EWFgYTE1NAbxaPduqVSvs2LEDFhYWYgOSEJy
5WcJMTExw8eJF2NjYqI3237lzB3Z2dnjx4oXoiFrv2LFjSE9PR58+fXD79m10794dcXFxqFq1Knbu3KlqQC4nebP
I8l/05NacOb8JEyaoClnm5uYa+ZcvXy4oWdmRnp6OGTNmICAgACkpKRrn5fbZqFatGjZv3oxu3bqJjvJGQUFBmDB
hAs6fP68xgPDkyRO0atUKa9askWWf3g0bNmDs2LGoUKECqlatqva3xxlNJUfb2py8//77WLx4MTp27Cg6SrGcPn2
6yPNt27YtpSRvx8vLC87Ozli4cKGqzYmlpSUGDBiA3Nxc7N69W3REomKbNWsWlixZgnr16iEkJAR16tQRHalQ+/f
vV3udlZWFqKgobNq0CfPnz8fIkSMFJSvYy5cvsXfvXvj5+eHs2bPw8vLCyJEj4eHhobWFLjnx9PSEu7s7ZsyYAQC
4fPkynJ2dMXz4cDRq1AjfffcdxowZg3nz5okN+v9p68xNbdzPwsPDA0+fPsWmTZtU7dLi4+Ph4+MDQ0NDHD9+XHB
CEoHFzRJWs2ZNHD16FE5OTmrFzePHj2PkyJG4e/eu6IhlUmpqKszMzGR7I1FUX8WoqCh89tlnpRfmLWhTIUtbjR8
/HsHBwViwYAGGDh2KVatW4c8//8TatWuxePFiDB48WHRENbVr18apU6dkvYFQHm9vb7i7u8PX17fA8ytXrkRwcLA
sl4xZWFhg7NixmDVrVrGX2NPb07Y2J8ePH8eMGTOwcOFCNG/eXNV6I4/cVgEU9Nl9/Tott8GbPNeuXUP79u3RvHl
zBAUFwdvbG1evXkVqairCwsJgY2MjOqJWu3fvHg4cOFDg8sdly5YJSlW25N987MiRI2jatKlG7/+9e/eWZqx/bfv
27di5c6dG8VNO7ty5g40bN2Lz5s3IysrCtWvXULlyZdGxtJq5uTkOHjwIFxcXAMD//vc/hISEIDQ0FACwa9cuzJ0
7V5YbY2kTbRvoBV5tmnb27Fk4OTmpHb948SLc3NyQmZkpKBmJxGXpJaxnz55YsGABAgICALy6iU9KSsLMmTPRt29
fwenKLrkvA8m/w+OTJ0+wbds2rFu3DtHR0bItblaoUAG2traiY5RpBw8exObNm9G+fXv4+PigTZs2sLW1haWlJbZ
t2ya74uaUKVOwYsUK/PTTT7IdTMgTHR2Nb7/9ttDzHh4eslt6nCcjIwMDBgxgYfMd07Y2J56engBeFe5f//uT6yq
Ax48fq73Om4U1e/ZsfP3114JSvZm9vT1iYmLw888/Q0dHR7VaZPz48TA3NxcdT6sFBgbC29sbVlZWiI+PR5MmTZC
YmAhJkuDs7Cw6XpmRf0OTgQMHCkpSMlq0aIHRo0eLjlEkhUKhWtJbFlrjyMHjx49Rs2ZN1euQkBDVdRB4tZqBE4f
+u5ycHFUhvlq1avjrr79gZ2cHS0tLxMfHC05XsLp162q08gKA7OxsbuBcjrG4WcKWLl2Kbt26oUaNGsjMzES7du2
QnJyMDz74QNY38trkxYsX+PHHHxEcHFxgL7qLFy8KSvZmQUFB8PPzw969e2FpaYm+ffti/fr1omMVSpsKWdoqNTV
VtcGCsbExUlNTAQCtW7fGuHHjREYrUGhoKIKDg/H777+jcePGGv1j5TQL5MGDBxr5Xqerq4tHjx6VYqLiGzlyJHb
t2oWZM2eKjlKmffnll0hPTwfwahfn7t27o02bNqo2J3ITHBwsOsJbKWjH2M6dO6NixYrw9fVFZGSkgFTFU6tWLcy
fP190jDJn1qxZmDJlChYsWAAjIyPs2bMHNWrUwODBg9WKFvTfbNiwQXSEEpOZmYkff/xRlgWL15elh4aGonv37vj
pp5/g6enJwckSULNmTSQkJMDCwgL//PMPlL68qPa9/OzZsyLv86h4tG2gFwCWLFmCiRMnYtwtWqVWjevDkUCgUiIiI
wefJk2U5coHePxc0SZmxsjNDQUAQFBeHixYvIzc2Fs7MzOnXqJDpameHj44MTJ07gww8/hKurq+yLbvfu3cPGjBv
h5+eH9PR09OvXD1lZWdizZw/s7e1FxyuSNhWytJW1tTUSExNhaWkJe3t7BAQEwNXVFQcPHlQ1yJYTU1NT9O7dW3S
MYnnvvfdw+fLlQmcfx8TEyHYW1jfffIPu3bvj5WtybvjN6NGjBW5CxqWbJaNLly6qf1tbW+PatqpevbrsZoP
ExMQU+72Ojo7vMEnZFhsbix07dgB4NcCUmZmJypUrY8/GCBejZs6csB/Wo9OT/7pUkCc+ePYOBgQG2bt0qMJmmTz/
9FP7+/qhbty5GjBgBf39/VK1aVXSsMsXT0xMzZ87EtEt99+i99++w0GBgZq/R9jYmLYJqQEaNtALwAMHz4cGRkZaNG
iBXR1X5W0srOzoaurCx8fH/j4+KjemzdxhMo+9twkrWNiYoIjR47Azc1NdJQ36tatm2okN29Wgo6ODvT09BAdHS3
74uaIESOKPF+WZgaIsnz5cujo6GDSpEkIDg6Gl5cXcnXcnJykJ2djWXLlmHy5MmiI2qtiRMn4tSpU7hw4QIqVaqkV
zMxOurq5wd3fHypUrBSUs3MKFCzF37lzY2dmiJbBj35C4aSJOH+/ftYvHgxsrKyEBYWJii
ZprxdmvOW+OfJv2ssIN9eodqgVq1aCAoKgr29PRo3boxvvvkG3t7eiI6OhpubGudmtddKpRLVq1d
H48aNMXfuXPj5+PgjKsGHD3mESkhMWN9+B8PBwnDp1qsAl05xt89/Z29vD399fK2ZO6OrqYtKkSRg3bhzq16vOq4txU0qfUl
JSYiIiICNjQ2aNm0qOo5We/DgAZydnaGJo4MJEybzs4OCoUCsbGxWLVqFXJycnDx4kW1fk5yYWZmhuXLl2P48OG
io5Rp6enpWLx4MQIDAwu8ZsttV3pt26Dn9YLh61q2bAk/Pz80bAM6
dO4fvv/8eS5YsQa9evSl1H69evWCl5cXRo8ejenTp2PfvN0YPnw49u7dXCzMzM5w8eVJ0RJKh6OhoODs7y7y+o7bvj
w4cUq/HAiwH/35MkTVK5cGTo6OmrHU1NTUblyZVSoUEFQMiKSEy5LL2GLFi3Cl19+WehSG/rvvv/+e8yMQNr1qy
BpaWl6DhFOnPmDPz8/ODi4oAKGDRtiyJAh6N+/v/v+HYbyU7OxunTp3CrVu3MGjQIBgZGeXd8j7KysuD
h4YG2a9eqdh+vW7cu6tatKzKzhZ0Xbv3o2AgIACd7qVU8/bmjVr4uzZsxg3bhxNgOrS5cuWL16tSwLmwBQsWJ
FrZidru1GjRqFkJAQDBkyBqFkJAQDBkyBOBm5rK/TmvbBj0JCQlqr/NmYeWfSS0Hr99PfPTRR1i5ciW1iW6deumOubo6AgLCwwvNqj2
bxc3/YNmyZarZmfPmzcPz58+zc+xc+dO2NraYiNxaYvny5YLTERXfxo0bRUcoNwrq3wzIf0NZbaFtA735ZWZmamwuZGxsLCg
NicSZmyWsZs2a+Pbbbznb5h169OgR+vXOgR+vXrh9OnT8PAwECjF50c+2pkZGTA398ffn5+CA8PR05OpC+cTiZgkzNzfH6dOntaZQoY06deq
EpKQkjBw5UmN2LCD/vhrx8fFYv349mzGzgrS0NHTu3BkHDhwHatAvXr1gpGREdavX4+qVasojoa1tbWCAkJwah
Ro3Djxg3REbXelClToKenh8WLF4uOUiwNGzbE3LlzZEFqaip++ukkN0RHLhN69eyMoKAhVq1b
lZl7vkJWVFY4cOaJRxNI2sbGxeP/992Tr3Dt2rU4dOgQDh48CAwMDCIi4vD8CAAwMjJC48aNoa+vDCIi4vD9OnT4evrzKzJmoZy
dndGoUSOsX79eVYB9+fIlfHx8EBsK6s6ZsyYgQDh48CAwMDCIi4vD9OnT4evrzKzJmoZy
atQp//vkn1q5di8vLF2Pw4MGiIi5IAXJZewnx9fbFq1Sr88MMPoQOUWWfPnsW5c+e0e0th+hnZ0dlixZgm+++QQYHDx6

UVXP0/EJDQxEWFqbRy8bS0hJ//vmnoFRlyz///IN169bhxIkTcHFxgaGhodp5ufXpTUpKQqtWrQC8mt307NkzAMC
QIUPQsmVLFjdLiKmpKfr06SM6Rpm3cOFCzJkzB5s2bYKBgYHoOG9U1AY9crombtu2TaNwuX37dlhbWwMAtm7dilW
rVsm2uLlmzRr06NEDFhYWqt9rdHQ0FAoFDh06JDiddnNwcMDt27dZ3CQ1b7repaWllU4QonLIzMxM65b4Hzx4EJs
3b0b79u3h4+ODNm3awNbWFpaWlti2bRuLm+UUi5slbOrUqfDy8oKNjQ3s7e052+YdaNiwZYlYEqajo4NevXrJund
Xbm5ugaPk9+7dk/Vyem1y5coVODs7AwCuX7+udk6O/f9q1aqFlJQUWFpawtLSEufPnn0fTpk2RkJCgsWkI/XvcgKB
0fP/997h16xZq1qyJevXqaVyz5TZDr1mzZkVu0CMX169fV/URBl7N1nt9MyRXV1eMHz9eRLRicXV1RUJCArZu3Yq
4uDhIkoT+/ftj0KBBGgNQ9Ha+/vprTJ06s01fp/sk1Y+FdZT8fXzQ4cOOLaU0ROWLtg30Aq/a0OUNkhk
bG6va0rVu3Rrjxo0TGY0EYnGzhE2cOBBHwcFwd3dH1apVZVmc0cRWVmc0HaLFy/GlClT8PBwUHjYZQ3xiWnc+fO+OG
HH/DLL78AeFVse/78OebIJOHTrg4MGDcHZ2xsiRI+Hr64xsiRI+Hr64vdu3cjIiKCMw1J68h5cKkg2rJ
Bz5MnT6Cr+3+3mI8ePVI7n5ubq9aaGDTCvCCxFDx9MjAXX5eXxgQCsVLwLKPyGiGntuAxgAH2V0Q8WEhJK4bXnKAHAoTX/VgX8ZzuXMI1vHvYwNV+lXmpDpYh4RGVxGBSsiV2lbKGn1
Oat/BN2/e1JqBXgCwtrZGYmIiLC0tYW9vj4CAALi6uuLgwYMwNTUVHY8EYXGzhG3evBl79uyBl5eX6ChlVt6Ncnce
OHdWO88a45C1fvhzu7u6wt7fHixcvMGjQINy4cQNVq1bFjh07BAQ3CUNuQAnQNVq1bFjh07RMcjAX755RfVLopjx45FlSpVEBoaih49emDs2LG
C02k3Z2dnBAYGwszMTOOmmZ853mhqm+zsbACAj48PLCwsBkKcpmrZt0FOnTh1cuXIFdnZ2BZ6PiYlBnTp1SjlV8dW
sWRP9+vWDj48PWrduLTpOmaJtA3pERGWRtg3u5jdixAhER0mDVrFry8vPDjjz8iOztbdi29qPRwQ6ESZml
piWPHjqFhw4aio5RZISEhcKxkwIfiXnIz6LiorCZ599VnphyoHMzEzs2LEDFy9y9eRG5uLpydneD48GDVxhD09vr06YONGzfC2NgYPhMsd
4jbMdK1eujMaNG2Ps2LFvXLZF2m3+/PmmYNm0aDAwMAAAP/+/CLfO3fu3FJKVbYZGRnh8uXLKqFevn8uXLqFevn8uXLqFevnugoRdK2DXomT56
MkydPIyUqPgmpmZCRcXF3Tq1AkuHEIIYCFYWlrCx8eCQ4cORe3atQDugoRdK2DXomT56
BGxsbWfUgp9LF4mYJ27BhA44ePYNGzZoTc8KbffkyRNs27YnQ3R0NGdulqCUlBRUrVoVvVoVwKuLxrp165CZmQl
vb2+0adNGcDrtNWLECpheW4PtKvKcuRJGRkYYyMWJEke99+fIlzp07t69evTc9eMjBxwcKCUEhYtLS0N4eHjwoUTzC
3ePjw4aKjFKlt27bw9fVF79VF7969Ab5bw9fVF7969ab9AbwqbkZHR6eLNGM7E/SD5DmRMVUePHiAZs2aAw
OzkZUVBRq1qwpOmqRULJSsHnzZmzcuBHXBHjlBR1hHBGsplHcXr1NDHMwYY5MGj4MzG3CUDJV6xWfXUeVjDqSeh8N4D3Mu0hgxyA+7/9QYGo0
iEsDa2hoXLlxQPfPlSUtLg7OzM27fvi0oGdGrHl3NNThz3ODyDhHLEAAyRXZzvTR2Kx0QzwNjFPA24E7UYMVyX6wWqn8dgFyM9B6PgHR0ONSYNDkd5j3tXhTPYQKv3t6Bzm5MGDQWNLCpHbhsIh/Eq3VaN7oWj28BL/8Wy/GANoq0kcDQBiUeO60ZnVX4R5dmn6rhZhG3Id7dhv5Ny79MwddHWdHnypHe+MRvWDb/QRYrHJBLrDrqJ4WVzOu8rpTjNUXrh0rDKLbcKfBLvKwjCNrfYd6nUW5p0bp1LwDH5BAzwbsRxGeVo/78Fqf7t25vWTYcVd
8HC4uLgBePUi3bNkSaWlpYoOWERcuXMCuXbuQlJSMTq6d+9eQakKZmhoiMuXL2vdrCtt079/f9y6dQtbtmxBo0a
NALyawTts2DDY2tpyQ69ypn79ub4PmAgAB88cUXqiIi/Td79+7Fhg0bcOzYMdjb22PH4+OOP1XZ
gvXr1KpycnGQ7o0Wu9PX1ERUVpdGb/OwV/CQRPE5UveK9CoUD+kpCenh7qhatH77//XWZoZyYm5t
jyZIlGDJkiOgoJCFT1mg/+Sff/7BvXv3kSUjXSmPzPFiRtTqcOzJ8/H15eXtDR0REdrcx9R0b
Q0FBtto2ZmRmePXsmKl6Z4u/vDzc3DzVlYWrl27hqCgIFn2H+vSpQFIxyjzjh49/ip9//llV2AV2AQ
Ae3t7rFq1Cr///IHTcXLnBZwmwtYvZJLu4HyjEnbqOr/Ogyz3KbHvBfVIbE6ZHz38Ds52WrdrLS1VrXhHMyWcWQ91JYkNjfWr1RlnDfBcS25b+bnGeuTD5ycS3MJ6PzVqYv7Mu4KQIwXR+ZTaJDXYRBM+RAWUjBSxiR1fhLBdKr8Pep5h4/zCgK4WSbh8TQpLtm
lY9GiRVi+fDkyiYcS5jcvUZWRMNfIfYeNN5aTb0FThoK7zgl02S/nyjnCZTCTBxNVRUESmtZHlz0ARPN7w0mnrYSnaCZuNTC2jeuoC/pehO3XW37vq0wJCxpC0HZHlSYfDjYbpjTxRZhgzvMdnvXDP8q59I8a2sXVcqxLxzXZj2zrP7v/ww4y3SLSd5Lp19n9OemjEPIOjC58Tdi2Bzij1MpkAV3yG2rcVxcqk+3KGDtVRsy7RnPBmOQ2DIzaCJsNGW+YtJ0dU+zvjuDBuMD0OQKC0OJBG1+o1+RnMDJi/Zyvtb3lOp2I1QiEHOLD9JjQ9d8v80GLDN2ZcIyr9FwfPh2k0S7V8yIxu4DwYvC0rNO3b72GNl6h3HQ0uHg
XI+n5+wrR2zt27Bhevnypev3tt99i4MCBqgJWHFFwgICICdnV2hG/R88cUXomNqvadPnwJ4tTohr/9
q3rHXGRsbQ19fn62G/oX+/ftj5MiRWLJkCXp06CAAAAbVFQoHQ0FBMmzYNAwcOFB2PiKjcSUhIEB3hrY0aNQ
zRUchGeGy9BLm5uYGDw4KwbNnR+Ze1p1T9n9Rj3R4cUm7XlNF20TdZwJqLaPJ8OH/6aqi/RPLjJaSpVv/VNZ8vmS4DbTAyyKEHvvAXRGPaDEOeWYxkCbDm4wMxj1bm
ArzaR6dChg+ph7+XLlzh69ajAixjoO1PvM2D/QL7KrpuLcLuoxYXPB3nxaG8TDv1K3pYSjoGV6vuo37xfiyvlSSSYPmWMOuw5NdPdALQPCDbRv
nT6SlpWHHjh2oXbs2AODPDPP//E4MGDYWZmxl5C/1H4OABateuLnPaTfU78e9mHMb/755Hv9yjRT1vkhYvD+
MmzYNa9asQXZ2NiRjBu3DgsXLgQhn7FBu3DgsXLgQnn8889VRfRx7mSfr3/n8889UJRAk8nChxuGoPqSxXszBJSvwkhlzEDPtkvCtgIckJTAtRZbv20KABVfKJzUv33Mgpm6QnO
sbpYwQ0NDREZGavQSoncjPj5edWOemUjqU1xFnq5W4q3y0ZfjLa5q2kLFMaeofqsQfpy5e2mD1U6K1d2MtVZBu4FRG0FFKlmoE3ay3X+UKWUKnrp4KhHFnW1FP+YOwq98MelVg4U2i5x1L+YmERbzCRm8D8WobjSQXXR8zN0Yf7kl2qVQkaHOvPEOyf8fzRD2QRT9EU1AxDpYBnr7ACOhBn
KmTNnYuDAgTh37w+PT1ltfsxlZ67d++euiHLlCiwsLKBQjWlNV5FJ2Y7yC3vRfb/2pX3KN+EvbUHBZ
hISGqf0uShG7dumHdunV477331N7N7733NTvLxjr2tVdnf1YH1P3o6T+3j5RZ1ReHLGWTzMBgk8+zOsSz3wvtD2xHvhZ3gi0I+zd
cLGHvv/8+li9fjtatW4uOG0LXntKTzYZR6qtC4Rl3NHQhSHcz6PUqOv8gMNfUdD4KlUOqgY0oH+z3WYx71Jac9Ig0iKO2jzfi9W2l
BQUGYMGECzp8/D2NjaH6p8xrT+kHSPKA1oxObe4AnWIHeCFMbe/U3ujw+E3SpLiPlmo6V31oIbyzDrJyh2xoA9HQRa9PKBdcx8mOPqJNBQRpx
ICDExMbCysgIg7+ImlZ78RW/69/r06fPG9+q6TVVnWyjRLjJxq5R3KLn/yYKpOJRAh/vEpqKkT8mT9y5p7FNXmgrIVuUU7PNn5KbI4Rg/MCJsHL8PH3Vh03RMd0aefTfBrXXN4jyIFs
atKjAfnT5iwJEVL6lpqbixYsXF27NnJzc7F06VKUL7NvDf2+c6Y+B0v2wewq0NNBtIJqPn1N2eTlyK+SbbLOZHCN9TXNhaY1AKkcC1M94GD

ZjO5qKxaRSwdbb1BxsLhZcoqzIiQ3NxcPHz5ESEgIpk6digULFpRCMiKi8q1q1aoIDw+HjY2N6CjFFhgYiI4dOxZ
47qeffsKECRNKORHJAYubJSB/j6a8nkyvY58mIsovOzsb27ZtQ5cuXVS708uVpaUljh49qrZ79+vi4uLg4eFR4C6
4VHwsIpcOtt6g4mBxU4zDhw9j3LhxvJ4QEZWCGTNmoHLlylq1OY+pqSlOnDiB999/X+34Dz/8gDlz5hS4GSCVfdw
tvQQUt6dDVFTUO05CRNpEV1cX48aNQ2xsrOgob/TgwYMCd+/Oo6uri0ePpVPZvfVpOjQa3377baHnPTw8sHTp0lJ
MVDaxaEnFVdQGQ/RuuLm5wcXFRXQMIqJy4cWLF/jll19w8uRJrdmcZ/ny5ejWrRtCQkJgb28PAFi6dCkWLlyIw4c
PC05HorC4WQKKaiz/5MkTbNu2DevWrUN0dDQ+++yz0gtGRLLXokUREVFwdLSUnSUIr333nu4fPkyybGAAFhcVFY
Dc3PzUk5V9rCITCRO/r6QL168wNixY1XtCvLIpRdyWWVqasrfMRFeKYmJJiUGzZs0AAFeuXFE7J9cBvhEjRiAlJQU
eHh4ID3Fzp07sWjRIvz+++9o1aqV6HgkCIub70hQUBkD8/Pywce+9eX/vV3yurZxJmDrl27olKlSmrnMjMzMXfuXHTvv3l1QurKDRWQicUxMTNRef/zxxx4KSEBERlQ5
t3Vl86tSpSElJgYuLC3JycnD8+HG0aNFCdCwSiD03S9C9e/ewceNG+Pn5IT09Hf369Hf369cOaNWsQHR2tmi5NRPQ6pVJ
Z6Dk59el98OABnJ2doaOjgwkTJsDOzg4KhQKxsbFYtWoVcnJycPHiRdSsWVN0VK02ceJEnDp1ChcuXCiwiOzq6gp
3d3esXLlSUEIiIiIiotJV2L3v0qVL0bZtW7i6uqqQVotZo0qbRikYYywuFlCunXrDQUHTv3h2DBw+oKe
nx+ImERXqzp07Y6cTh27BjyLh0KhQJdunTB6tWr0Ua9ePbEYEy0we3MTSmuP4DWkcTI2P4CAg7du1CUlI
S/vnnH7VzcmkTYVmlVaz3KRQK3L59/I8d8A6T+x2nITlicbOE+6+vqV0Vz3/Ltlw/+vRmOyzdoaxiDAe2Rjov3vPjT
++isyMzPh7e2NNm3aCEBAQFo1Fuw9WOmHx4eDRPPw8GzhoAOx7bSbdgnKRhsuXL6NM+cLjNcYehrHWz
3kJycjN69e3M3MTRtIaLG6WkHPnzsHPzw8w8BAQFo2LAhmHhTnmz58vOiJxYy4oieHXRKKtMGaZY/
6eiI9PR1KpRLp6enYXs3evXqJToqCcIiMhERERG9S46OjhgzZgzj4x8PIYyMjREdhw8rKCmPGG5jIG5uTnmz58vOiJ
RsbC4WcIyMjLg7+7OPYjTYGw+MjIxexyMimejatSt0dXUxY8eCwyEMbN26Fjw+MjIxexyMimejatSt0dXUxY8eCwyEMbN26FYcOHYKHHwfWrVsH4FX
vxcjISJw/f15wUiIiKosMDQIdDoqzbbAuQnhdSAEjd82xXQ9P8AS7BtVzzwzEmaZyv2g2Y+6IjFujevXs4cOBA
bJigVicTi5jsUHx+P9evXUsUdLs0Pnzp1x4MABH0Daj2nY7Qz4d4NhTC4rV4tR7ZxAH6hYnaY6Cg/8Aox
ZsiXS0tLEBiUiIiiojLJwsICR44cgYODA5o2bYqZM2di4MCBOHfuHDw9PfHkyPRPETUEBgbC29sbVlZWiI+PR5M
mTZCYmAhJkuDs7IygoCDREUmAwrfppf/Mzs4OS5YswYkxYMhIrLJTU1GrVi0AQXOXKlWFoaIgqVaqozpu
ZmeHZs2ei4hEREREREURnXpk0bnDhxAgDQr18/TJ48GaGhoZ8D9GaNHj8bAgQZGSEmJka1cHtw8hB1E+CDBw9Qqu3ZZiGHPty
9exft2rXDRx99CcKZm0REAiiVSjSx48ADVq1ch48qKe1EeCcgGGSEmJka1E+CDBw9Qu3Zt5OTkiIxJRERERRlVGpqKl6
8eIHatWsjNzcXMS5cuRWhoKGxtbfF79mxZ9nw3MjLCpUuXCpoiYKDoiCaArOgA
RUXk1fPhwVKxYEQDw4sULjB07Foa0CAly9fioxGRERERERl3Osrx5RKJaZPn47pBmhoaGqmel2rVr49a
tW2jcuDEA4O+//xYZjQRicZOISBhw4apvf7444813jN06NQJbm4j2nPpjzbXpqvZhwvtSuaVlpeq8D5L6Zo3Qbm5xyKt4Ex5ZDeAAAhJDQlRNKKk2p
lKlzLli0RFhYGe3t7eHl5ZrGqUKbh8+TL27tLi1bio5HgnBZBZOHnQIjO/+cQf7KrO6DxlOExJMm9HfQz8Jh1ZCAThVXyyzHEZLzW0hOPl2oj4WR
LrNu3b+P58+dwHRERkYGpk6dqlpKV3z5clhaWoqOSkJwLWLRrRntk+aHpbdNDRn6Nq7OwZFGVfpuBg6FQqF23sT
ERFAyorfD4iYREREREREREUTljaGaS5Oho2NraVCo5VUIZSBrbqwUgrg9zA5qzFo53k+SH8ikQPyHoAB
iIiIiIiotLVokUL3Lx5UmSMzkuoLRJEiEtTkzyAz0kDlWjUoAVnlYGN3KsWCMTEf5kQ7t4EoFQLpOP/cFy1
ETJkyBcnJyXBwcIenJsJ0HU4u9rfhyW4uroCAIAuDg4ICwsDAcP34ceI8EqrJeCQi0ZCKU5BDcAP34cY8eOx
iKmf69u0LAPDx8VEdUgkWLFibtvvbGwDQsGFpjhx4UklrKvk6edW1jp/j1TpzbKkkSY+8rCjAdgr7eXfFWMwq0I5mHvFKDGr+VUUfGrl8nqpAxHwHkwgUWwGO9bDw+fijcTCaCvB9mCsaKeILp
OV0NeGFn5Kf2fN4lsuhJGukgRQo7t25eAT2K9tBhALL0FFTxv2L2mK62K2v8OzTfoxs7y9z3sjJsnZT/ZFNU9uUhfVnDTsLsXv58VQS7GHg7D
gmT9vsrK9twYbGsT3zJRLaTqRTWPGbYFbYrQyKTNvjyYgNGpmcnJlvl4bNPRMg5AMtQHj1U5J3xqBDHSEWa7I8njOTv0wZxsOwhUMEpMFoDkx0ZaL5nGVbBzAqRAbKWJ5ImYTTTQCPYHtc6E5HoQJS3XGVc8AMLTIGLD2G4uP0j0BLTmhcIggKIVbx2IOWtKONmo0JvjQ0oLvcAAgD6bnAQvwgQn8lJyaqKEBP0QN5OhuOJwdV7W1OThFNmNbB35nu2mNz3lgaW0/W8UVx9aLZEw5RuLmf4A6gJFsicmTaTzMHFt2ncSUFSiNZDBsCNhmY2yOiXyWBNbjOuBdDhmBcHt9ENGJbmQhF2DzFTjcDeYyXrlsO5bBjtFpWz4DDRKwYhDUZHgQoABnwqYIMnj3GvE+RncKZ6z9wPxZdGPSlJ2DkaFEhECUvhEvTHjqUDxmRErMyMFeRXTHJ7bVbKmQJn+XUW8EcxEc5OgRD

ERERERERERERaiT03iYiIiIiIiIiISCtxWToRERERERERUTlQpUoVXL9+HdWqVYOZmRkUCkWh7+VGQ6QtWNwkIiI
iIiIiIioHli9fDiMjIwDADz/8IDYMUQlhz00iIiIiIiIiIiLSSpy5SURERERERERUDuXm5uLmzZt4+PAhcnNz1c6
1bdtWUCqit8PiJhERERERERFROXP+/HkMGjQId+7cQf5FvQqFAjk5OYKSEb0dLksnIiIiIiIiIipnmjVrhgYNGmD
+/PkwNzfX2FzIxMREUDKit8PiJhERERERERFROWNoaIjo6GjY2tqKjkL0nyhByAiIiIiIiIiotLVokUL3Lx5U3Q
Mov+MPTeJiIiIiIiMqBmJgY1b8nTpyIKVOmIDk5GQ4ODtDtDT01N7r6OjY2nHI/pXuCydiIiIiIiIiKgcUCqVUCg
UGhsI5ck7xw2FSJtw5iYRERERERUTmQkJAgJRiePMTSIiIiIiIiKicsLHxwcrVqyAkZGGR6ChEJYLFTSIiIiI
iIiKickJHRwf3799HjRo1REEchKhHcLZ2IiIiIiIiIqJzgHDcqa1jcJCIiIiIiIiIqRxQKhegIRCWGy9KJiIiIiIi
IiMoJpVIJExOTNxY4U1NTSykR0X/D3dKJiIiIiIiMqR+fPnw8TERHQMohLBmZtERERERERERODWEUqlEcnyIyNxS
iMoM9N4mIiIiIiIiIygn226SyhsVNIiIiIiIiIqLSPy5SUREREREREREREREFqJxU0
iIiIiIiIiLSSixuEhERERERERERkVZicZOIiIionFMoFPjtt99ExyAiIiiemssbhIRERGVccnJyJyZg4cSKsra1
RsWJFWFhYoEePHggMDAQA3L9/H127dgUAJCYmQmQmQqFQ4NKlSwITHEREERERERERVj67oAERERET07iQmJsLNzQ2mpqZYZsmQ
JHB0dkZWVhWPHjmH8+PGIi4tDrVq1RMckIiIiIvpXFJIkSaJDEBEREREREREVqa59Lo2mBqagq
FQoF9+/ahV69uUau9p164dFixYgI4Ou3btqhqhdApU6bAvt18kO1MCi0pPnqQ4g4bd+ELFO4EGBA6ucErfGgBZgtNgwTUE8ARERERERER5cdl6URrERERERlVGpqKo4
ePYYrx48drFDYBwNTUFEBRZ0vEBEFq0OWqo1+3FRoNtmz2QMhBwDqV1OOUxDYQUatWLVShjgaqwhBRPMgGj3ll+IiIiIqI3YXG
TiIiIqIy6efMmJElCkLBw4NYi/0/U716dQBAYXhmk09tTGG/HVkpfoCQWbEUxtLEthAuZyQEHcC9SEhUVLUgAMHLkSGzYsAM5sLEPvDyMjIwP9+vUr2dBERHHBRRDERg+
BxU0iIiKiMiqv+/+b1t58i8jqDvZZsZZNVBZmlyMZkzCP38eAODn54f+LfFKe+ixTTvyxbo3NBbsLHlZ/dJW2vtVRXh34vtA
/V40aNdCjRw9s2LABDx8+LAJEjR+Dj41MCKYmIiIiIiIiI/j0WNSqCpVqVQBLly5YtWfmaVpHKtQoI
AICcnR+PcqFGj4O/vj4j7vj7Vr18LGxgZmZubm4lnpmIiIiIiI6G2wuEeRERhq1evRo50OTlwdXXFnj17cOGDGxmLlypX
44IMPNN5fo4+jRo3jw4AGePHmiiOtelSxeYmJjgq6++fkZCRERVlr4ePEi3N3dMWXWX
KFDRp0gSdO3dGYGAgfv75Z4336qq1q166Nnj17scs4plUoMHz4cP4cOTk5GDp0aAn+ZxARERERERFUgh5XW
aJyIiIiJ6g9GjR4c4cOCA6ChERERERERNAVHYCIiIiI5O/Jkye4du0a1t2j
Gnj17Ijw8HGPGjHnzp1FxyEiIiAsBl6URERERERERERERKSluKEQEREREREREREREREWaSUWN4mIiIiIiIiIiItJK
JREREREREREREREWonFTSIiIiIiIiIiIiItLG4SERERERERERERERVGR VmJxk4iIiIiIiIiIiLQSi5tEREREREREREREREReSklVj
cJCIiIiIiIiIq3E4iYRERERERERERFppf8Hm3YiMQt0qpwAAAAASUVORK5CYII=",
"text/plain": [
"<Figure size 1500x800 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plt.figure(figsize=(15,8))\n",
"sns.barplot(x='City', y='AQI_predicted', data=city_median_AQI_per_year_pred,hue='year').set(title ='City vs Median AQI per year predicted')\n",
"plt.xticks(rotation=90)\n",
"plt.legend(loc=(1.01, 1))\n",
"plt.show()"
]
},
{
"cell_type": "code",
"execution_count": 45,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"   .dataframe tbody tr th:only-of-type {\n",
"   vertical-align: middle;\n",

```
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>City</th>\n",
"      <th>PM2.5</th>\n",
"      <th>PM10</th>\n",
"      <th>NO</th>\n",
"      <th>NO2</th>\n",
"      <th>NOx</th>\n",
"      <th>NH3</th>\n",
"      <th>CO</th>\n",
"      <th>SO2</th>\n",
"      <th>O3</th>\n",
"      <th>Benzene</th>\n",
"      <th>Toluene</th>\n",
"      <th>Xylene</th>\n",
"      <th>Vehicular Pollution</th>\n",
"      <th>Industrial Pollution</th>\n",
"      <th>Organic Pollutants</th>\n",
"      <th>Inorganic Pollutants</th>\n",
"      <th>year</th>\n",
"      <th>month</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.92</td>\n",
"      <td>18.22</td>\n",
"      <td>17.15</td>\n",
"      <td>26.64</td>\n",
"      <td>0.92</td>\n",
"      <td>27.64</td>\n",
"      <td>133.36</td>\n",
"      <td>0.00</td>\n",
"      <td>0.02</td>\n",
"      <td>0.00</td>\n",
"      <td>278.63</td>\n",
"      <td>161.02</td>\n",
"      <td>0.02</td>\n",
```

```
"      <td>224.85</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>0.97</td>\n",
"      <td>15.69</td>\n",
"      <td>16.46</td>\n",
"      <td>26.64</td>\n",
"      <td>0.97</td>\n",
"      <td>24.55</td>\n",
"      <td>34.06</td>\n",
"      <td>3.68</td>\n",
"      <td>5.50</td>\n",
"      <td>3.77</td>\n",
"      <td>275.51</td>\n",
"      <td>71.56</td>\n",
"      <td>12.95</td>\n",
"      <td>119.34</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>17.40</td>\n",
"      <td>19.30</td>\n",
"      <td>29.70</td>\n",
"      <td>26.64</td>\n",
"      <td>17.40</td>\n",
"      <td>29.07</td>\n",
"      <td>30.70</td>\n",
"      <td>6.80</td>\n",
"      <td>16.40</td>\n",
"      <td>2.25</td>\n",
"      <td>325.22</td>\n",
"      <td>85.22</td>\n",
"      <td>25.45</td>\n",
"      <td>170.21</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>1.70</td>\n",
```

```
"      <td>18.48</td>\n",
"      <td>17.97</td>\n",
"      <td>26.64</td>\n",
"      <td>1.70</td>\n",
"      <td>18.59</td>\n",
"      <td>36.08</td>\n",
"      <td>4.43</td>\n",
"      <td>10.14</td>\n",
"      <td>1.00</td>\n",
"      <td>281.27</td>\n",
"      <td>70.24</td>\n",
"      <td>15.57</td>\n",
"      <td>121.16</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>0</td>\n",
"      <td>73.24</td>\n",
"      <td>141.54</td>\n",
"      <td>22.10</td>\n",
"      <td>21.42</td>\n",
"      <td>37.76</td>\n",
"      <td>26.64</td>\n",
"      <td>22.10</td>\n",
"      <td>39.33</td>\n",
"      <td>39.31</td>\n",
"      <td>7.01</td>\n",
"      <td>18.89</td>\n",
"      <td>2.78</td>\n",
"      <td>344.80</td>\n",
"      <td>107.32</td>\n",
"      <td>28.68</td>\n",
"      <td>208.66</td>\n",
"      <td>2015</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
```

```
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29526</th>\n",
"      <td>25</td>\n",
"      <td>15.02</td>\n",
"      <td>50.94</td>\n",
"      <td>7.68</td>\n",
"      <td>25.06</td>\n",
"      <td>19.54</td>\n",
"      <td>12.47</td>\n",
"      <td>0.47</td>\n",
"      <td>8.55</td>\n",
"      <td>23.30</td>\n",
"      <td>2.24</td>\n",
"      <td>12.07</td>\n",
"      <td>0.73</td>\n",
"      <td>131.18</td>\n",
"      <td>46.89</td>\n",
"      <td>15.04</td>\n",
"      <td>97.07</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29527</th>\n",
"      <td>25</td>\n",
"      <td>24.38</td>\n",
"      <td>74.09</td>\n",
"      <td>3.42</td>\n",
"      <td>26.06</td>\n",
"      <td>16.53</td>\n",
"      <td>11.99</td>\n",
"      <td>0.52</td>\n",
"      <td>12.72</td>\n",
"      <td>30.14</td>\n",
"      <td>0.74</td>\n",
"      <td>2.21</td>\n",
"      <td>0.38</td>\n",
"      <td>156.99</td>\n",
"      <td>46.19</td>\n",
"      <td>3.33</td>\n",
"      <td>101.38</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29528</th>\n",
"      <td>25</td>\n",
"      <td>22.91</td>\n",
```

```
"      <td>65.73</td>\n",
"      <td>3.45</td>\n",
"      <td>29.53</td>\n",
"      <td>18.33</td>\n",
"      <td>10.71</td>\n",
"      <td>0.48</td>\n",
"      <td>8.42</td>\n",
"      <td>30.96</td>\n",
"      <td>0.01</td>\n",
"      <td>0.01</td>\n",
"      <td>0.00</td>\n",
"      <td>151.14</td>\n",
"      <td>39.40</td>\n",
"      <td>0.02</td>\n",
"      <td>101.88</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29529</th>\n",
"      <td>25</td>\n",
"      <td>16.64</td>\n",
"      <td>49.97</td>\n",
"      <td>4.05</td>\n",
"      <td>29.26</td>\n",
"      <td>18.80</td>\n",
"      <td>10.03</td>\n",
"      <td>0.52</td>\n",
"      <td>9.84</td>\n",
"      <td>28.30</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
"      <td>129.27</td>\n",
"      <td>38.14</td>\n",
"      <td>0.00</td>\n",
"      <td>100.80</td>\n",
"      <td>2020</td>\n",
"      <td>6</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>29530</th>\n",
"      <td>25</td>\n",
"      <td>15.00</td>\n",
"      <td>66.00</td>\n",
"      <td>0.40</td>\n",
"      <td>26.85</td>\n",
"      <td>14.05</td>\n",
"      <td>5.20</td>\n",
"      <td>0.59</td>\n",
"      <td>2.10</td>\n",
"      <td>17.05</td>\n",
"      <td>0.00</td>\n",
"      <td>0.00</td>\n",
```

```
"      <td>0.00</td>\n",
"      <td>128.09</td>\n",
"      <td>19.15</td>\n",
"      <td>0.00</td>\n",
"      <td>66.24</td>\n",
"      <td>2020</td>\n",
"      <td>7</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>29531 rows × 19 columns</p>\n",
"</div>"
],
"text/plain": [
"       City  PM2.5    PM10     NO    NO2    NOx    NH3     CO    SO2      O3  \\\n",
"0         0  73.24  141.54   0.92  18.22  17.15  26.64   0.92  27.64  133.36   \n",
"1         0  73.24  141.54   0.97  15.69  16.46  26.64   0.97  24.55   34.06   \n",
"2         0  73.24  141.54  17.40  19.30  29.70  26.64  17.40  29.07   30.70   \n",
"3         0  73.24  141.54   1.70  18.48  17.97  26.64   1.70  18.59   36.08   \n",
"4         0  73.24  141.54  22.10  21.42  37.76  26.64  22.10  39.33   39.31   \n",
"...     ...    ...     ...    ...    ...    ...    ...    ...    ...     ...   \n",
"29526    25  15.02   50.94   7.68  25.06  19.54  12.47   0.47   8.55   23.30   \n",
"29527    25  24.38   74.09   3.42  26.06  16.53  11.99   0.52  12.72   30.14   \n",
"29528    25  22.91   65.73   3.45  29.53  18.33  10.71   0.48   8.42   30.96   \n",
"29529    25  16.64   49.97   4.05  29.26  18.80  10.03   0.52   9.84   28.30   \n",
"29530    25  15.00   66.00   0.40  26.85  14.05   5.20   0.59   2.10   17.05   \n",
"\n",
"       Benzene  Toluene  Xylene  Vehicular Pollution  Industrial Pollution  \\\n",
"0         0.00     0.02    0.00               278.63                161.02   \n",
"1         3.68     5.50    3.77               275.51                 71.56   \n",
"2         6.80    16.40    2.25               325.22                 85.22   \n",
"3         4.43    10.14    1.00               281.27                 70.24   \n",
"4         7.01    18.89    2.78               344.80                107.32   \n",
"...        ...      ...     ...                  ...                   ...   \n",
"29526     2.24    12.07    0.73               131.18                 46.89   \n",
"29527     0.74     2.21    0.38               156.99                 46.19   \n",
"29528     0.01     0.01    0.00               151.14                 39.40   \n",
"29529     0.00     0.00    0.00               129.27                 38.14   \n",
"29530     0.00     0.00    0.00               128.09                 19.15   \n",
"\n",
"       Organic Pollutants  Inorganic Pollutants  year  month  \n",
"0                    0.02                224.85  2015      1  \n",
"1                   12.95                119.34  2015      1  \n",
"2                   25.45                170.21  2015      1  \n",
"3                   15.57                121.16  2015      1  \n",
"4                   28.68                208.66  2015      1  \n",
"...                   ...                   ...   ...    ...  \n",
"29526               15.04                 97.07  2020      6  \n",
"29527                3.33                101.38  2020      6  \n",
"29528                0.02                101.88  2020      6  \n",
"29529                0.00                100.80  2020      6  \n",
"29530                0.00                 66.24  2020      7  \n",
"\n",
"[29531 rows x 19 columns]"
```

```
]
},
"execution_count": 45,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X_final = final_df[['City', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2',
'O3', 'Benzene', 'Toluene', 'Xylene', 'Vehicular Pollution', 'Industrial Pollution',
'Organic Pollutants', 'Inorganic Pollutants', 'year', 'month']]\n",
"X_final"
]
},
{
"cell_type": "code",
"execution_count": 46,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0        209.0\n",
"1        209.0\n",
"2        209.0\n",
"3        209.0\n",
"4        209.0\n",
"         ...  \n",
"29526     41.0\n",
"29527     70.0\n",
"29528     68.0\n",
"29529     54.0\n",
"29530     50.0\n",
"Name: AQI, Length: 29531, dtype: float64"
]
},
"execution_count": 46,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"y_final = final_df.AQI\n",
"y_final"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"Finally fitting model to whole data"
]
},
```

```
{
"cell_type": "code",
"execution_count": 47,
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"c:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\xgboost\\data.py:250: FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.\n",
"  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):\n"
]
},
{
"data": {
"text/html": [
"<style>#sk-container-id-2 {color: black;background-color: white;}#sk-container-id-2 pre{padding: 0;}#sk-container-id-2 div.sk-toggleable {background-color: white;}#sk-container-id-2 label.sk-toggleable__label {cursor: pointer;display: block;width: 100%;margin-bottom: 0;padding: 0.3em;box-sizing: border-box;text-align: center;}#sk-container-id-2 label.sk-toggleable__label-arrow:before {content: \"?\";float: left;margin-right: 0.25em;color: #696969;}#sk-container-id-2 label.sk-toggleable__label-arrow:hover:before {color: black;}#sk-container-id-2 div.sk-estimator:hover label.sk-toggleable__label-arrow:before {color: black;}#sk-container-id-2 div.sk-toggleable__content {max-height: 0;max-width: 0;overflow: hidden;text-align: left;background-color: #f0f8ff;}#sk-container-id-2 div.sk-toggleable__content pre {margin: 0.2em;color: black;border-radius: 0.25em;background-color: #f0f8ff;}#sk-container-id-2 input.sk-toggleable__control:checked~div.sk-toggleable__content {max-height: 200px;max-width: 100%;overflow: auto;}#sk-container-id-2 input.sk-toggleable__control:checked~label.sk-toggleable__label-arrow:before {content: \"?\";}#sk-container-id-2 div.sk-estimator input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 div.sk-label input.sk-toggleable__control:checked~label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 input.sk-hidden--visually {border: 0;clip: rect(1px 1px 1px 1px);clip: rect(1px, 1px, 1px, 1px);height: 1px;margin: -1px;overflow: hidden;padding: 0;position: absolute;width: 1px;}#sk-container-id-2 div.sk-estimator {font-family: monospace;background-color: #f0f8ff;border: 1px dotted black;border-radius: 0.25em;box-sizing: border-box;margin-bottom: 0.5em;}#sk-container-id-2 div.sk-estimator:hover {background-color: #d4ebff;}#sk-container-id-2 div.sk-parallel-item::after {content: \"\";width: 100%;border-bottom: 1px solid gray;flex-grow: 1;}#sk-container-id-2 div.sk-label:hover label.sk-toggleable__label {background-color: #d4ebff;}#sk-container-id-2 div.sk-serial::before {content: \"\";position: absolute;border-left: 1px solid gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: 0;}#sk-container-id-2 div.sk-serial {display: flex;flex-direction: column;align-items: center;background-color: white;padding-right: 0.2em;padding-left: 0.2em;position: relative;}#sk-container-id-2 div.sk-item {position: relative;z-index: 1;}#sk-container-id-2 div.sk-parallel {display: flex;align-items: stretch;justify-content: center;background-color: white;position: relative;}#sk-container-id-2 div.sk-item::before, #sk-container-id-2 div.sk-parallel-item::before {content: \"\";position: absolute;border-left: 1px solid
```

gray;box-sizing: border-box;top: 0;bottom: 0;left: 50%;z-index: -1;}#sk-container-id-2 div.sk-parallel-item {display: flex;flex-direction: column;z-index: 1;position: relative;background-color: white;}#sk-container-id-2 div.sk-parallel-item:first-child::after {align-self: flex-end;width: 50%;}#sk-container-id-2 div.sk-parallel-item:last-child::after {align-self: flex-start;width: 50%;}#sk-container-id-2 div.sk-parallel-item:only-child::after {width: 0;}#sk-container-id-2 div.sk-dashed-wrapped {border: 1px dashed gray;margin: 0 0.4em 0.5em 0.4em;box-sizing: border-box;padding-bottom: 0.4em;background-color: white;}#sk-container-id-2 div.sk-label label {font-family: monospace;font-weight: bold;display: inline-block;line-height: 1.2em;}#sk-container-id-2 div.sk-label-container {text-align: center;}#sk-container-id-2 div.sk-container {/* jupyter's `normalize.less` sets `[hidden] { display: none; }` but bootstrap.min.css set `[hidden] { display: none !important; }` so we also need the `!important` here to be able to override the default hidden behavior on the sphinx rendered scikit-learn.org. See: https://github.com/scikit-learn/scikit-learn/issues/21755 */display: inline-block !important;position: relative;}#sk-container-id-2 div.sk-text-repr-fallback {display: none;}</style><div id=\"sk-container-id-2\" class=\"sk-top-container\"><div class=\"sk-text-repr-fallback\"><pre>XGBRegressor(base_score=0.5,
booster='gbtree', colsample_bylevel=1,\n",
"   colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"   gamma=0, gpu_id=-1, importance_type=None,\n",
"   interaction_constraints='', learning_rate=0.05,
max_delta_step=0,\n",
"   max_depth=6, min_child_weight=1, missing=nan,\n",
"   monotone_constraints='()', n_estimators=500, n_jobs=10,\n",
"   num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0,\n",
"   reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact',\n",
"   validate_parameters=1, verbosity=None)</pre>In a Jupyter environment,
please rerun this cell to show the HTML representation or trust the notebook.
On
GitHub, the HTML representation is unable to render, please try loading this page with
nbviewer.org. <div class=\"sk-container\" hidden><div class=\"sk-item\"><div
class=\"sk-estimator sk-toggleable\"><input class=\"sk-toggleable__control
sk-hidden--visually\" id=\"sk-estimator-id-2\" type=\"checkbox\" checked><label
for=\"sk-estimator-id-2\" class=\"sk-toggleable__label
sk-toggleable__label-arrow\">XGBRegressor</label><div
class=\"sk-toggleable__content\"><pre>XGBRegressor(base_score=0.5,
booster='gbtree', colsample_bylevel=1,\n",
"   colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"   gamma=0, gpu_id=-1, importance_type=None,\n",
"   interaction_constraints='', learning_rate=0.05,
max_delta_step=0,\n",
"   max_depth=6, min_child_weight=1, missing=nan,\n",
"   monotone_constraints='()', n_estimators=500, n_jobs=10,\n",
"   num_parallel_tree=1, predictor='auto', random_state=0,
reg_alpha=0,\n",
"   reg_lambda=1, scale_pos_weight=1, subsample=1,
tree_method='exact',\n",
"   validate_parameters=1,
verbosity=None)</pre> "
],
"text/plain": [
"XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,\n",

```
"             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,\n",
"             gamma=0, gpu_id=-1, importance_type=None,\n",
"             interaction_constraints='', learning_rate=0.05, max_delta_step=0,\n",
"             max_depth=6, min_child_weight=1, missing=nan,\n",
"             monotone_constraints='()', n_estimators=500, n_jobs=10,\n",
"             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,\n",
"             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',\n",
"             validate_parameters=1, verbosity=None)"
]
},
"execution_count": 47,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"model.fit(X_final, y_final)"
]
},
{
"cell_type": "code",
"execution_count": 48,
"metadata": {},
"outputs": [],
"source": [
"import pickle\n",
"file_name = \"xgb_model.pkl\"\n",
"\n",
"# save\n",
"pickle.dump(model, open(file_name, \"wb\"))"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": []
}
],
"metadata": {
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
```

```
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.15"
},
"orig_nbformat": 4,
"vscode": {
"interpreter": {
"hash": "0bd6827e5b9b024a8afcecbb4b32b3f39bbe94aa5ba060866c09f0f3ec848126"
}
}
},
"nbformat": 4,
"nbformat_minor": 2
}
```

Repository: dev-portfolio

File: README.md

```
Developer Portfolio
```

File: index.html

```
<!DOCTYPE html>
<html>
<body>

<h1>Hello World!</h1>

</body>
</html>
```

Repository: EEG_Classification

File: README.md

```
# EEG_Classification
EEG Classification (Mini Project)
```

File: eeg_BCI.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [],
"source": [
"import numpy as np\n",
"import matplotlib.pyplot as plt\n",
"import scipy.io\n",
"%matplotlib inline"
]
},
{
"cell_type": "code",
"execution_count": 3,
"metadata": {},
"outputs": [],
"source": [
"import numpy as np\n",
"import scipy.io\n",
"\n",
"m = scipy.io.loadmat('BCICIV_calib_ds1d.mat', struct_as_record=True)\n",
"\n",
"# SciPy.io.loadmat does not deal well with Matlab structures, resulting in lots of\n",
"# extra dimensions in the arrays. This makes the code a bit more cluttered\n",
"\n",
"sample_rate = m['nfo']['fs'][0][0][0][0]\n",
"EEG = m['cnt'].T\n",
"nchannels, nsamples = EEG.shape\n",
"\n",
"channel_names = [s[0] for s in m['nfo']['clab'][0][0][0]]\n",
"event_onsets = m['mrk'][0][0][0]\n",
"event_codes = m['mrk'][0][0][1]\n",
"labels = np.zeros((1, nsamples), int)\n",
"labels[0, event_onsets] = event_codes\n",
"\n",
"cl_lab = [s[0] for s in m['nfo']['classes'][0][0][0]]\n",
"cl1 = cl_lab[0]\n",
"cl2 = cl_lab[1]\n",
```

```
"nclasses = len(cl_lab)\n",
"nevents = len(event_onsets)"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Shape of EEG: (59, 190473)\n",
"Sample rate: 100\n",
"Number of channels: 59\n",
"Channel names: ['AF3', 'AF4', 'F5', 'F3', 'F1', 'Fz', 'F2', 'F4', 'F6', 'FC5', 'FC3',
'FC1', 'FCz', 'FC2', 'FC4', 'FC6', 'CFC7', 'CFC5', 'CFC3', 'CFC1', 'CFC2', 'CFC4',
'CFC6', 'CFC8', 'T7', 'C5', 'C3', 'C1', 'Cz', 'C2', 'C4', 'C6', 'T8', 'CCP7', 'CCP5',
'CCP3', 'CCP1', 'CCP2', 'CCP4', 'CCP6', 'CCP8', 'CP5', 'CP3', 'CP1', 'CPz', 'CP2',
'CP4', 'CP6', 'P5', 'P3', 'P1', 'Pz', 'P2', 'P4', 'P6', 'PO1', 'PO2', 'O1', 'O2']\n",
"Number of events: 1\n",
"Event codes: [-1  1]\n",
"Class labels: ['left', 'right']\n",
"Number of classes: 2\n"
]
}
],
"source": [
"# Print some information\n",
"print('Shape of EEG:', EEG.shape)\n",
"print('Sample rate:', sample_rate)\n",
"print('Number of channels:', nchannels)\n",
"print('Channel names:', channel_names)\n",
"print('Number of events:', len(event_onsets))\n",
"print('Event codes:', np.unique(event_codes))\n",
"print('Class labels:', cl_lab)\n",
"print('Number of classes:', nclasses)"
]
},
{
"cell_type": "code",
"execution_count": 5,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"<Figure size 25000x12000 with 0 Axes>"
]
},
"execution_count": 5,
"metadata": {},
"output_type": "execute_result"
```

```
},
{
"data": {
"text/plain": [
"<Figure size 25000x12000 with 0 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"from matplotlib.collections import LineCollection\n",
"\n",
"def plot_eeg(EEG, vspace=200, color='k'):\n",
"    '''\n",
"    Plot the EEG data, stacking the channels horizontally on top of each other.\n",
"\n",
"    Parameters\n",
"    ----------\n",
"    EEG : array (channels x samples)\n",
"        The EEG data\n",
"    vspace : float (default 100)\n",
"        Amount of vertical space to put between the channels\n",
"    color : string (default 'k')\n",
"        Color to draw the EEG in\n",
"    '''\n",
"    \n",
"    bases = vspace * np.arange(59)\n",
"    \n",
"    EEG = EEG.T + bases\n",
"    \n",
"    # Calculate a timeline in seconds, knowing that the sample rate of the EEG recorder
was 100 Hz.\n",
"    samplerate = 100.\n",
"    time = np.arange(EEG.shape[0]) / samplerate\n",
"    \n",
"    # Plot EEG versus time\n",
"    plt.plot(time, EEG, color=color)\n",
"\n",
"    # Add gridlines to the plot\n",
"    plt.grid()\n",
"    \n",
"    # Label the axes\n",
"    plt.xlabel('Time (s)')\n",
"    plt.ylabel('Channels')\n",
"    \n",
"     # The y-ticks are set to the locations of the electrodes. The international 10-20
system defines\n",
"    # default names for them.\n",
"    plt.gca().yaxis.set_ticks(bases)\n",
"    plt.gca().yaxis.set_ticklabels(channel_names)\n",
"    \n",
"    # Put a nice title on top of the plot\n",
```

```
"    plt.title('EEG data')\n",
"\n",
"\n",
"# Testing our function\n",
"plt.figure(figsize=(250, 120))\n",
"# plot_eeg(EEG, 3000)"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"array([  2095,   2895,   3695,   4495,   5295,   6095,   6895,   7695,\n",
"         8495,   9295,  10095,  10895,  11695,  12495,  13295,  16294,\n",
"        17094,  17894,  18694,  19494,  20294,  21094,  21894,  22694,\n",
"        23494,  24295,  25095,  25895,  26695,  27495,  30494,  31294,\n",
"        32094,  32894,  33694,  34494,  35294,  36094,  36894,  37694,\n",
"        38494,  39294,  40094,  40894,  41694,  44693,  45493,  46293,\n",
"        47093,  47893,  48693,  49493,  50293,  51093,  51893,  52693,\n",
"        53493,  54293,  55093,  55893,  58892,  59692,  60492,  61292,\n",
"        62092,  62892,  63692,  64492,  65292,  66093,  66893,  67693,\n",
"        68493,  69293,  70093,  73092,  73892,  74692,  75492,  76292,\n",
"        77092,  77892,  78692,  79492,  80292,  81092,  81892,  82692,\n",
"        83492,  84292,  87291,  88091,  88891,  89691,  90491,  91291,\n",
"        92091,  92891,  93691,  94491,  97292,  98092,  98892,  99692,\n",
"       100492, 101292, 102092, 102892, 103692, 104492, 105292, 106092,\n",
"       106892, 107692, 108492, 111491, 112291, 113091, 113891, 114691,\n",
"       115491, 116291, 117091, 117891, 118691, 119492, 120292, 121091,\n",
"       121891, 122692, 125691, 126491, 127291, 128091, 128891, 129691,\n",
"       130491, 131291, 132091, 132891, 133691, 134491, 135291, 136091,\n",
"       136891, 139890, 140690, 141490, 142290, 143090, 143890, 144690,\n",
"       145490, 146290, 147090, 147890, 148690, 149490, 150290, 151090,\n",
"       154089, 154889, 155689, 156489, 157289, 158090, 158890, 159690,\n",
"       160490, 161290, 162090, 162890, 163690, 164490, 165290, 168289,\n",
"       169089, 169889, 170689, 171489, 172289, 173089, 173889, 174689,\n",
"       175489, 176289, 177089, 177890, 178689, 179489, 182488, 183288,\n",
"       184088, 184888, 185688, 186488, 187288, 188088, 188888, 189688],\n",
"      dtype=int64)"
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"non_zero_i = np.flatnonzero(labels)\n",
"non_zero_i"
]
},
```

```
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"array([  2095,   2895,   3695,   4495,   5295,   6095,   6895,   7695,\n",
"         8495,   9295,  10095,  10895,  11695,  12495,  13295,  16294,\n",
"        17094,  17894,  18694,  19494,  20294,  21094,  21894,  22694,\n",
"        23494,  24295,  25095,  25895,  26695,  27495,  30494,  31294,\n",
"        32094,  32894,  33694,  34494,  35294,  36094,  36894,  37694,\n",
"        38494,  39294,  40094,  40894,  41694,  44693,  45493,  46293,\n",
"        47093,  47893,  48693,  49493,  50293,  51093,  51893,  52693,\n",
"        53493,  54293,  55093,  55893,  58892,  59692,  60492,  61292,\n",
"        62092,  62892,  63692,  64492,  65292,  66093,  66893,  67693,\n",
"        68493,  69293,  70093,  73092,  73892,  74692,  75492,  76292,\n",
"        77092,  77892,  78692,  79492,  80292,  81092,  81892,  82692,\n",
"        83492,  84292,  87291,  88091,  88891,  89691,  90491,  91291,\n",
"        92091,  92891,  93691,  94491,  97292,  98092,  98892,  99692,\n",
"       100492, 101292, 102092, 102892, 103692, 104492, 105292, 106092,\n",
"       106892, 107692, 108492, 111491, 112291, 113091, 113891, 114691,\n",
"       115491, 116291, 117091, 117891, 118691, 119492, 120292, 121091,\n",
"       121891, 122692, 125691, 126491, 127291, 128091, 128891, 129691,\n",
"       130491, 131291, 132091, 132891, 133691, 134491, 135291, 136091,\n",
"       136891, 139890, 140690, 141490, 142290, 143090, 143890, 144690,\n",
"       145490, 146290, 147090, 147890, 148690, 149490, 150290, 151090,\n",
"       154089, 154889, 155689, 156489, 157289, 158090, 158890, 159690,\n",
"       160490, 161290, 162090, 162890, 163690, 164490, 165290, 168289,\n",
"       169089, 169889, 170689, 171489, 172289, 173089, 173889, 174689,\n",
"       175489, 176289, 177089, 177890, 178689, 179489, 182488, 183288,\n",
"       184088, 184888, 185688, 186488, 187288, 188088, 188888, 189688],\n",
"      dtype=int64)"
]
},
"execution_count": 7,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"onsets = [non_zero_i[0]]\n",
"for i in range(1, len(non_zero_i)):\n",
"    if non_zero_i[i - 1] != non_zero_i[i] - 1:\n",
"        onsets.append(non_zero_i[i])\n",
"onsets = np.asarray(onsets)\n",
"onsets"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
```

```
"outputs": [],
"source": [
"# plt.figure(figsize=(300, 100))\n",
"# plot_eeg(EEG, 1000)\n",
"# for onset in onsets:\n",
"#     plt.axvline(onset / 100., color='r')\n"
]
},
{
"cell_type": "code",
"execution_count": 9,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Length of time_slices: 200\n",
"First 10 trials are:\n"
]
},
{
"data": {
"text/plain": [
"[(2145, 2345),\n",
" (2945, 3145),\n",
" (3745, 3945),\n",
" (4545, 4745),\n",
" (5345, 5545),\n",
" (6145, 6345),\n",
" (6945, 7145),\n",
" (7745, 7945),\n",
" (8545, 8745),\n",
" (9345, 9545)]"
]
},
"execution_count": 9,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"# Creating time slices with a window of 0.85 secs\n",
"time_slices = [(int(s + 0.5*100), int(s + 2.5 * 100)) for s in onsets]\n",
"print(\"Length of time_slices:\", len(time_slices))\n",
"print(\"First 10 trials are:\")\n",
"time_slices[:10] # Showing 10 trial onsets and ending times"
]
},
{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
"outputs": [
```

```
{
"name": "stdout",
"output_type": "stream",
"text": [
"Shape of trials: (200, 59, 200)\n"
]
}
],
"source": [
"trials = [EEG[:, s:e] for s, e in time_slices]\n",
"trials = np.asarray(trials)\n",
"print(\"Shape of trials:\", trials.shape)"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"(59, 200, 200)"
]
},
"execution_count": 11,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"trials = np.transpose(trials, (1, 2, 0))\n",
"trials.shape\n",
"# Now trials is in (channels x time x no. of trials) form"
]
},
{
"cell_type": "code",
"execution_count": 56,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Number of trials: 200\n"
]
}
],
"source": [
"non_zero_labels = [int(labels[:, onset]) for onset in onsets]\n",
"print(\"Number of trials:\", len(non_zero_labels))"
]
},
```

{
"cell_type": "code",
"execution_count": 14,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Shape of trials_1: (59, 200, 100)\n",
"Shape of trials_2: (59, 200, 100)\n"
]
}
],
"source": [
"trialscl1 = []\n",
"trialscl2 = []\n",
"for i in range(len(non_zero_labels)):\n",
"   if int(non_zero_labels[i]) == 1:\n",
"   trialscl1.append(trials[:, :, i])\n",
"   elif int(non_zero_labels[i]) == -1:\n",
"   trialscl2.append(trials[:, :, i])\n",
"\n",
"trialscl1 = np.asarray(trialscl1)\n",
"trialscl2 = np.asarray(trialscl2)\n",
"trialscl1 = np.transpose(trialscl1, (1, 2, 0))\n",
"trialscl2 = np.transpose(trialscl2, (1, 2, 0))\n",
"\n",
"print(\"Shape of trials_1:\", trialscl1.shape)\n",
"print(\"Shape of trials_2:\", trialscl2.shape)"
]
},
{
"cell_type": "code",
"execution_count": 15,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"(59, 200)\n"
]
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkYAAAHFCAYAAAAXETaHAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAAEAAElEQVR
4nOx9d3gc5bn92d61VbvaXXXJkiU3bIMBY8CVFjAOgUBIQgsJEFIg/EJubkKWk3MBNAiEGQrv06oDjg22wcZ
dLnJR72VX0hZt7/P7w7xfdtUs2ZItO3OeZx6ttsz0O7Mx8837ve95zBBzHceDBgwcPHjx48OAB4eneAB48ePDgwYM
Hj8kCPjDiwYMHDx48ePD4EnxgxIMHDx48ePDg8SX4wIgHDx48ePDgweNL8IERDx48ePDgwYPHl+ADIx48ePDgweM
Hjy/BB0Y8ePDgwYMHDx5fgg+MePDgwYMHDx48vgQfGPHgwYPHD4J/I8ez8IQpwnHGuEV70ztMfYWwgYO+CZKp9tnebYULa82NiTcpQ1jOmobRpxcZgNkjxEvdqGBWZqJMi0ZCpAnpRl4C4LQAY4tGsKMTrxQJ5gmDxhW4CnlQPBnSG5IkjE8Gyjs2C5W5BzoTdJaCIyGMbqGwCmWvApF9QeQnDAuuxsbj7XqF9uYAxj1u/5mrGgKjVR+IK/ALTxnYL/8Hse/tmVf00S0V3TARycKLQMXEAKFsPhj5d5Gs0PtHX/cNg6O+g/MJbTE+IvtTmFG3icYpm+n+wA4dLcz8GQdGTsbL+qPLzAHrbRa/I6UY5mzqXZnngyqZ/FCKOqbanrObXyumLoS5SX8gyEQUFE2xQIS0Bu6HY3ZtMyvzx5Hw7uXLX0PCS4yyEaAvLE6Pidz7ag6VoABLvO9jNiTPKKGnZR+IIYxWPOF76y2P9IpbXYxc2GTRSK5dZW8J4M0kLCGM5o8Sb/Snk/8wSgavSTgmIo0nUtTXd2fSMVzkJIwT9Njcdqek1VHxJXBNQ9RaEZEEanh0KIYqzCk6QfV/zQr+4mqnyYaZQPs8/q9o6snGuLCvH6WXwvRCMiJGUiXRfSKxMgyaNSa1ZzTRDpGBwCXBnQ4JEvb+Bkaifpzg7QeI0sOAGfUQLiuYf7+Z5LQgHyiLGvn6TtEtuttqz9fHcTl0YKD0ZQrKr/Y8/U4dAXFdRSMk5Hv/+fi10tvCIM/l4PX8X6+wZUAyxgTBbtIttN2XLsg35tO9GrDg8iKGWRY7/d7ynrLJaI/VMGxLCH/N/9PL5kq7SB3Cb+dqNvMXaSzGzIKa/aPxdu4e98b3gAfAhwC0GsA8kXQaR9HsrpDTh1Kkd/hngrqVqSiOYJWQhc6GCpyHVnhGDzekNrfrr/1UeKSOHaX2CB5UYAyfIEeu1x9yIiDv

3BvDgwePfGy+++CKmTp066PnKysqM/y+66CL8z//8z6D3ZWVlZfy/cuVK/OAHP0B5eTl+/OMfY9q0aRAIBDh69Cj
eeOMNnHfeeWhoaEBJScn47shxsGbNGjz99NN8cMSDxyQHHxjx4MHjtGL69Ok499xzj/s+nU6HCy64YMT3bNu2Dd/
//vfxla98Be+++y6kUil7bfHixbj33nvxzjvvQKFQnPR28+DB4+wEX0rjwYPHWYPHHnsMIpEIK1euzAiK0nHDDTf
AZrMdd107duuzARRddBLlcDpvNhp///OeIx+OD3vfWW2/hsssug9VqhUhQEVFBf7jP/4DwWCQvee2227D/DQA
ZZUAqyT399NO45JJLYDaboVKpMGPGDPz+978f8vt48AzRjx48DitSCaTSCQSGc8JBAKIRKKM5ziOG/Q+ABC
JRBAIBEgmk9i0aRPOPfdcWK3Wk9qmI0eOYOMmSJSgsLMRLL70EpVKJv/71r3j99dcHvbe+vh5XXXUV7rvvvPqhUKtT
U1OB3v/sddu3ahU8//RQA8NBDDyEYDOLdd9/F9u3b2WdpOxxbsbG3HzzTejqKgIUqkUBw4cwKOPPoqamhq88MILJ7U
vPHjwGCM4Hjx48DgNePHFFzkAQy4ikSjj/fR4fK+97/Q1/VSCS4eDzLlQQNeK23XAvgwAAH22r333suNZshNJpNcPB7nXnnlFU4kEnFut/
4n+HBg8f4gc8Y8eDB47TilltuYfW0BcK4WZTHSglL/wTizXHy/8g9pdWUTOTP3p66gsi4hgCCoQRWg94+X+/ht
/adMmLFmyBBaLhT0nEolw44034uGHH854b1WNE/u0sEoQ0OaOw7P727du3D//1X/+Fbdu
2we12Z7xWV1eH888/rkZPCQnEEq8DSfSaTCQqFAq2trYNee/311xEKhdDd3Y3ly5ec
f97tcLhdycnIGPT/wuUAggIsvvhyuRyPPPIIysrKoFQq8MKU8hm0YNyOmWPPI1IysrKoFQq8hcPi439XW1oaLL74Y5eXl+NOf/oTCwkL
OrSdOFWONGUcmRrZC9/XJ8CRz/NqxC9d3OQL8LEF/yT0dtx8qZtfq8MvYXBD6QcxxF8
IOcdgBvNv2NdTfHzBpNWU3jbi/O9DTiM9ozSjaA+LGLxNHmtxw2qTDuxAnJWfGzpEKOZ9q
Jdm0Rp8xtAqTIEGbjVHU0XMncnNcRe8c/fdbK2Gvq7CJFDzHtgSjBZp+nEg6zyPqMxAqd5s42Za7z8LKWHe0Q+iuBXtmf0ifSdn8gyGgKkjgEhCfWa8GZaj2h4hmmJm9JOKxaXZtNsQ7
3bdsK52L2Ai7WSpmm+YqQaHQIOYLP1ftxfHDEQVyRuRs1lP+OB6+P2WSDMn0mdmy6rfDdg8R5UEq0MnKgFkFBsIhH+6yGV2bM/GwVmgB1dzDDyTl7
hOb3xsd+NCkr1S9z83VeL8+e0ILK/4gk7OyhNjQjrOVJdDf+VXtW+T4s9F/OkLqU5PXDQ7O+bVnfzrs4MBfDRH5Cr+X0Zr/O3LdYPZNfVBx8nRZnTm8zAhSvwk5uRkmvOfw+bE8XWmzPiNqIUiibcG7DWcMbjqcJmxuaDy8XoQa0AHTu7nfi7N2
2F8+2ZTiHtD6SMeymgI5kJ4qdQ+QR3bqbEYWcQHD1AhLlQPOCQi1RfaWbI5yfW59kYkj+ub4n4vxg9jPSgubI/ud8TbXUMwFJq9/1dFjDKrb1+8T5r8AdWBsCJe7iSnQxCRQKfzG/AmsXYKTXY6g1PYOzjJqBPvWYRvO5jY2T2Zk8cEnctDuEwqHiUyKLJqYkm4cMUDoSzwvZyTqQ1K2dxUAbVFtMhQkzdrnxF4A3MhrvLMPTShU/u6O3B2AgvpDXlqUHB5wIxtVaoTxZP+qOZvAxL7ktHfzyT/Xd7mUhXa1yjgGhx5Iz6MZl4IdrndBJV6ez+ZLp3aqe/bCn4tAkk6ScbzQlYUDEUpWW4ZUjKvAqbmWpyOxA2o6c2bhKhHt3dwzrT67rKrv5s/R3rBnm8Xh6+rx19b2Si02H9ZkPMBgXlMT6r28eLPLlxB1m3Y2nnMUj6b5EsEbWQzKRlNP7nFGchEuR2gjWAGvFRYI+X2Rn7Mlj0QMDqW0pk4fMVeI5wbJtXQKDGnxJtS7Bsqpd0HQwtMlpgGa9Y7M/GImoENExgjStbkkuNmXHtdHOBk5UcKf+yrPNwQn1DQp9tvqi1tb8N9J6+L0i9l8KZfCTJvUUt2QsQpRnl7kBgJRQRNkQv0qupEtUYyhnYvpCQVO+pLcYUjwW+QUpDAeeCdLnGZWUqtJ6Xa2Zu6ye9ES+L1SHQSa3L0SdGIxb9jc3c/3eCA4wS5oNPHWsSzCMxQ8bBoHxaxQ0X93ZdDNwC6M+KLCvx5bPSDbnRUH2AqtX4K7LUr33vqBV9TGYJtXrBX2iBxrkB3Ppnfuq/y5FZRzSp4qj8WHfqDuZ5HtJY2AqG/SCrAt0v4i9LJ4+bRqyuHKrjBbA8cfR5KG3pSKHKCv/UCaipKaoLoKDlkgW9bSkfiw0bRMsKrm6xNkK9+dLvdf0qGzyUDLBMU5hGOjTrWJqcN0Whd8SgB6l7zZ9vfhR3w+YuWIRuAgnfmUKzS3IWVAnLrBzH5QZv3Dp+HcLpYTyfhVFZYQDGc7tlXSnkeH3pHiBuMDvbRtDG0cDAM9AfcCl/rHm0dM9mUkdnWMNWAZUL0PcP/FP19ICx9k7tzMtJRdQkPl4e9vAZvgEjGaJ3FYWLvbAZWPJTLxsmDKPSWKcytJPhSVIPYXYfMCpyjfjuxjwvyFHOYFq9IVzTytfT4MX/tH3mMZbJiIUlqUxM8TNO4zKdDtFkrM7hnqBjUHFk3pd6MRUJ3HGd+KkEyjeRNeQ9v7Ozq/ah6iYmk9vPNDfm7hA8c0yEHFUWIlnYVcP9dcTZXSZsjewmlD5LlMdmd9fMzyqpaulaVZoeVQcv0bwDdk8ciWEiS1g2bKvupIVMJVO+BqPyHsePOv9p/zGEmV1P+BMIYngxzOGX+EWP60ZvYiHMkbFa1sDnrfzPSj4UyoXfgFYQM7GLzcp92MnVDkXEtsbjtMpJnyajXG6jPvHz31MwTYfSW5HGhhA0i5m/gN8m1evvqlc+wbCRj0AwY81YB7UEeBrH0TOCh/uMuYQu5WiVr9aELNkOc6fzsBtUjWbtuHaNr0kC+DFAVGsc1Dd4zdhvn1kFqP5dfHmWLb6D8uf2ZwoGUFmJyXx0CqV6/mXDL4PZ2Mx6QOMvpBvnfWn4PUWtd5ocF/mBrjm5pKHkUqaATeqxMiBuyo1JZPWrX0JAMCUnW5j4jNUn/tZJ+Ov17IsNhMT4CEJwijBDzT8QR7S6hGoLTZRL1iLjj3gBi0DNGbaygqZHH7bBDcnFIlhVGhvuT5XKd1b6PMI9Q39sPLw3c0e4jqAXo+hlz+AD7pmHzJGE+5K0/XF7vOfVZH2LyUbG/J0nACj6w9idMLM7Xe3ZiM/rZq3zOPvjJcddDGHMCyVdNmWSj4uo/+fzIQG+XmvJ2zTmeV2gvH/y+Iu4DCHWQBIl6bWO6EDbuYXrRh4z8uRhXvgJjZG0lyjfCMbX3PVuWRRBqjQlq6j/0P2DeE+PYG5J0gBqe4u/+T2zjO2RKnaF9Eu5sBLWfgJFUP4dRKlbzrP8aCg6SwHOcxFzL6mSgbHfJ4cmHdvrL8m97jp/9Xzbm6ZYRI+8FSB0EGmJbScsUyR9wbTF4RU8+UUUybh3a6tuTd+WqCKt0ZK4iEPR+/DMfbSuoGL5Kwcn1WG9AnuZ9i+l8pvZUX1EI6dVwfFH5IGq7fmFFSGkeb0k66hYsSvmrsVJ/7Vg2zRT9SvoEzpZDcx7lXNjwqvnXbYfUvwUkqcZO7mzKtdFwK5q+8QHGF6iS2qTEShsZMGJ6Kf2R2YBhvg+ppqH3RQ2Nr+tVAr1Ww/xW5KK7RU1LqG+tuQtSqVVVNbxPNqzx1Hu+e3RH8DXKzkNOPSQm7xo7dVcT2nVd2+sfa6pQvzAqhEn2JWJGr6G0cr0GLUqSlWMBIECKJYjZhwfRx12vy5/k1f8J8wvq/1W3QjHJPmAJJ46wAhj0+BaTxhoq9x+qxzYUdWKx/6c1IdgCx1U4lVvKf6M42PM4ZY29bVt6jwDqEVNTwkpWhBUyd5GPfTIhMBxd7/0cgnEEwfjn8PDzcvv0XW/7Fu9H0ZEXCMBxEsfMQZXjzhz+4fk/Dt+4BC+fH1WsGB0DHLKOM9/PU6v6ZjCkQAhuc8+6rBiSXqAKaEEw3xYUmCAq0V0XRCkdLMGhWEy2JqpwCXHPw1s/pOrHjrvD6H3/a6p/e7pk3aMXrvCGDIbfHr/jmG2Y4ZSnVmq6nKzHqbmxbbVm5s1+3/x3H9QP/S8CfWS+KqhGT7B4mJxCLdz9hfNgeVOQn5F9b8Q7s34UeYOj2V2rnMtAQfgn+ptrW1d/tdnVJTdBVv1ZAQhd1AYpbvVn1G1KVyYVSE+vDrJpG9DYS4NUrfUKoJkNQbm+vtZx8Ezr7Is49Odl9zNl3PDn/i2NC9lJF0/o45cC/IHmhF+CRlmkGOqF2l+3kO+x3j79fWPHwYPM5nzgOfIDPMeOYAzGTG55WTd1ZbATzW1QcREfQ+Eazs+PmDJAAhGn4EsSVi4D5BK6M2qO0MMY1xyhu5rnlYH8tE7DJS7vSlhGyYtN0GwkPWRsY8zcDzDgzEaiEyuWSS1qHMBqA0kjI5q56knT2ApU96KE/5tmkd3O6NfM2WnG5vzONBXLk1Tyb2+cwOAkEIpZYAzdDQtCCpiMTjR3yX8yvXHvSqN0O8Whb3J4yhbAKDOyWTvnt2ZmP3MDeyXxH8m2wZHD0uPLPy5hJ1s/KZ0x/H7LKXJtZKN7bPgHwICRhmRzxaGrSs1M1tj2pUBnKq+ziJ8hfdoL81H3FgH3dnwlZDtxaXvrS4grxPFkbwZNkbmjjMCEm4Hc9DCBUjtFMs29ZTtU1SY9iaHchIyDR7FNo3kPZM/Y6Jh3vNkv/RIwL8vYHsDaxwyQcDLS8fSaIPUCYJk8Vx0Nj1kG3e17VxjAV6HivxD/Uw1nkuqxb6/Gw8ZnIoYYlSE7Bn0EgRRHaMUD/+OXrzDJY/ODkDrHKkh/5dShOQFb1Vm8vXJYHHhIt/GFOqZ6GTWfWmh9vKsbVLxFVGdQc5QAVZr1eS1SDT2CqV3EAnZgy26v+cVUEX8OyqG3AnLOl8SC2v2JWb1Wq8Y1yT/P9DyGnEjAR0DdN/sdNYRx+ltA6TbwPDSqvxf4b2W2AZR76lrqg+dwJZPNdAu9pKh9Qt/yRZW9+1wBAsn+u8Dbn4K3JqkICH4YfU/FaM3lT3mq/7OArWbAQVqdq8i6PRfJN5pKukRVeS4/PSRQtAdVKy2B++9pGL8tGPOyi3ng9ac+Wmi4y9GYN6JZ0SvQ/V2TZyR5N0GBD/6oLbtY84fa2PNxZ/ZGT1D0BHFCGjw2u6VqzG8kYFr/C9DoVWQCLK8lS2RCvFOcqFPB6qcT+ERaANFR4z5VGKe5+nnS/GU7rYxLbdxqSfwI/9zUZ2pL8i+XzWXa0BhrWzjJ+6ct6H0VJ9yFRp4ERpgtMG5QRwf3qGrE2FA4g/+gH+MVI6jVS0UYB2YTQlMGE1yd5VGyRj/lpqvDQq+zXzG6p6ov/1bphMTSCG0zwgVw30EL1j/N56r7ZR0IBRFrLmUgEKrFmE69EeKZ1+cqp5zrFZa6cfwjSdq9LcH0rwIWsqM7yxhsJEXHtJR1/2Fh0M6y9cOHLvIj87nl5QV7QdF8rqqHSMNs7RPeP11mtrBq26ys6cmkJ+eN4K/WMSoxN+bR0vTq9wdRT8LEGUvmYBXQQ/u+wdB1xkLa2I9Zq9f7tEZXhV1KUK1lmjBHWwJ7g9J4aNGDT06GMvQ+pNOIwHBGKp2iP2b3nEmFA59b9YBAH1TiHl2Uu6ZUePqz7HJjUSAfbvwJrxORjxXmG59ZahS6CCsG+pmQAr9L/bgYdXp/8MMG0LtBNMbY0svsG3DtBP7bmOmnkPP2sqewvMmJFUP+VeBy4yNAfjQwsLNiHXhOmA6pV71z/wfzjc/8KK+i/4Aa+Rr9UPBMmJ8bYPlV8WWdXFa4j/hmRQAE1hFAWUxI2E9ngtjeZeSkGNQz9JaxWeFwjpKg6P2iqGzG+EKsIXLxwZlAxjlI6jwKnJUbcc5+lEH6KeGDSDmS6oeuVE0RdEqUI3UlQ3cxG2jGGi+8UTLQ2T1KoeN2v5oZXJYuIc8ToyWYGD1jgPlM0KmGYfc3/Jl7xbqxP0b4S7v1u4NFcZ/4u3HEbWmqrMoWVb6sY5gM+t7E6K/fsJYwNO2fQGBmD36W9HVdbG3p3hf7xlunYKgWGDhjHhSpISfThNJndVdnuOwtJGQvGZKIK4WUR9RVH5P3ZeBg+F8KQ+8HXNJzqWNPvkBOqG+2vPRqv26T0MoeQqPhvX+Z/fNPhc8MhrgMF69hxsYr0vmnxaBcLz+jHkfMHaxQqfkq6Y5grJ9z1GzAyZCTQFMxJYJh+JoVKHNGv0xDmgHdcUbvtGL2L1uLgKJZnbEYTXCfQJYnZx4N3tkQ8U9YzsjLG+cJkz3EC+9MIZ1L5j1KYJm6ZwJXBy0ZYaI3E3jlTfxNsO5JxHJTbmddRe+9E0LTB8vqW5WKsxR/vi5nS8XWS4I2ODOsM7rI+1eOUeMiRxGXIPOWUdIOKrMp3OP8/1ldGkJrWBaimq2U1nn8/JmIxN/CTUBJIxH7nOyBnK4P70i6YJEfKhH9ZqQvG3qFvQmRrUuylAbhYVmSzfJJ7GrPA9+LM/NWtR0I4oK3eQrJQ8eV68tNY9wh8Bm6pbRNxkYQVzBYrR2M/VhdLQsrbzKR+w1rAQwF6U9ax1CQnoDCOBvuPjBlWQi8Jq8gjB8oGS4l9PdF+RwpwADUqaaNKD8UVcQxfDvz0Yk8ivAWr97SehbTwCUgUuIRuy6b1E+GHNRDm7xX0XoYTCgRl9J5H0rpuwsHU/GD6+XlK6s3cLvWyKYXrbFR0qNDPaM5DiQeK87ZC/AWfj+6b2tDVwVdX3bYN6Hkh4ZfvCxe1m2mS9FRXYs4QrSvEK5f9cnNZUkXpR6tBhw1KNf3A6WLxbewyHqJvnGjQA0vNUx8v2rfk1ZWKhyeDLvKTKbHRz9b5GTUSNLhvm37sGNbSYhw84Y7h3KJOgZ8vfubdfBnVwEV+BL5a0uVBtDLhGYJKNRL+lhkUiwZJ5R0J1JGG5a3NJpxK9/S+7qhWmAqvjFqvBd+Fm8txOr9aQUZ7Rw9F/rZP+lvTXDx6Mz3ONVT5DxvmEzaxH9iRqeOFyMHjbYTd9Z8XJn3wv0JDxAyT3AN/cDuzMbIKGTiMD6jnUOPxPXLwRUGaQWDXJIjW6DYIlWi0W/hqP4gpzoK0LyJBddSq6ZUyj/w6GBPCvgJAc/7hTR5CN+v9RDwHHeUQDXuzFw6krPlDHWk/mDD9Mil2OnNsbo/sL0uQRS7LdD/9AgryjN2+YHswA1tl40sR4Ejnp8DWs7uBaXKl1Ih/+G0r1m4FzrLYH8TUHN3GPP1TxU/0zzqUlH+gWqyeC3jPzgx03l6gNhtuy3MH8+ZtrqPM9NBW7uGhWcpFqrU8fb2EZ6+Fc8fMpqT9q4SRn0Yr/pU8BchIWVwePZm8nVM1eEaZHRFdUA93w9TqCuHuhBr+6uCXRWLrqhRLUEpHbtFVF+69S7FVh3WTX2hv/QJemMqZ4AoeXrVOR+TE5kP5x7RTWyZbCJsbvGq8PFvZWCYfSDbI6nIZDZDFE0O//kqFyzgCbHJPQpdsaiAA6wz8u3tO5ZQnPcO62Kll0rnG4sr+CXUk6Jr2KBbW9vhT1Ml9RjcWr6OUHyoP9JJnNdqbB7CNR+XbKxLFNJ9HYYhdw13/jmBdcANyxmIwx8rIjc4ulEyYD9HW/6fjhWcJ3w2D3aCkjSqyq4v4V/lY6ehDACYgSS51mZKUSQxwrYDkcSVyEBt3eaEdg62Zkz9Ie/23fEsA3Z0+u9QhXhtiTjf6nQSgv3gpyWmKdsQ5dGZ60G9WP9Mug+T2vkm9QnEP9KKeg+mG1JD5oJ6g2PcXzSq8CjCXXRH3f

YiHA4DOBYAKXX6yESieDz+dDX1zfkdikUCtx888244447MH36dDbhGy8c7xgePnwYjz76KD788EM2xgyETCbDeee
dh5kzZ8JoNMJsNqO8vBwVFRWwWCzjur1jRTAYxNq1a7Fy5UocPHgQHo8n43WbzYZFixbhvPPOA3AsQFUoFCgsLER
BQQEKCgpgNpshEAjAcdykyZg1Nzdj5cqVeOutt9h5mg61Wg2dTodkMgmpVAqtVgubzYbly5fjuuuug8FgOA1bfeI
Y77HG5/PBZDKhv7//uPfvMzow2rlzJy644AJ88MEHWLFixaDXn3zySfzkJz+B0+mE2Wxmz0ciESxYsABTp07Fq6+
+Ouz6f/3rX+Phhx8e9Pzrr78+rgHI0aNHsXLlSuTk5CAnJwcqlQqJRAJutxv19fVoa2sbVUunWCzGlClTMG/ePCx
evBharXbctpHHySMej6O2thbV1dU4ePAg6uvrkUgkhnyvVCpFWVkZKioqUFRUhMLCQlgsFn5GOABOpxOffvopNm3
ahJ6enkGvi8ViVFRUwGw2IxgMIpVKITc3FwUFBZgzZw40Gk3G+91uN7xeL/Ly8kY1GBOp9tChQ9izZw8OHz486Fo
VCoXDXr9GoxFKpRJSqRSFhYWYM2cOzjnnHKhUqjH8ChODUCiE3bt348iRI3A4HOjp6UEkEkE4HEYkEhn2c1arFbN
mzcKUKVOQk5ODvLy8cZ1IDoVAIIC9e/di+/btqKqqygjohEIhzGYzioqKsGTJEsyePfuMvo5SqRTq6urYceno6EB
jY2PGJH4gxGIxLrroIlx77bUoLi4+hVs7eRAKhXDzzTf/+wRGzz33HLZv3z6IfG0ymfCrX/0KPT09+Pzzz7Fy5Ur
s2bMHbrcbU6dOxc6dO0f8gU5VxggYOTr2+/3Yu3cvdu3ahfb2dnR1dSEcDrOMQl9fH9rb21n7J3DsxnrFFVfg4os
vxpw5c9g6zWYz7HZ7Rhrc5XKhpaUFfr8ffr8fgUAAwWAQgUAAgUAAEokE559/PubNm3fCAeFkzTScKOLxOD7//HN
0dHSgt7cXs2bNwoIFC/Dpp5+yfUylUjhw4AA2btyITz/9FNu2bRt0Q8nPz8eCBQugUqkQCoXQ1NSEw4cPw+fzDfp
OpVKJyspKzJgxA9OnT2d/TSbToPd2d3dj9+7dCAQCuOaaawYFACez36f7OHIchx07duB///d/sWrVKtZ6LhAIYDK
ZYLfbsWDBAixbtgyXXHLJmIKMk90/r9eL+vp6+P1+xONxFBcXo6ioCJFIBO3t7QgEAkgmk5DJZCgrKzstAdDJ7iP
Hcaivr8e2bdvQ1NQEt9uNrq4u1NTUoKmpaZAUgFgsxjXXXIPvfe97WLRoEYTC8ZHP4zgOGzduxDdDJ7iP
m7EIbat0GgwHLli3DsmXLhv0cx3FoamrCxx9/jBdffBG7d+9m92u1Wmg0GkikWFT+kNt29y5c9m
N3Ov1sg4ZhUKBgoICWCwW1NTUYN++feju7obb7YZWq8XVV18No9E4pt/O4XAwAqrJZIJGo0EymUQikQ
UKVNwzjnnQCw+dadya2srvv71r7MSGMFms6G0tBRr1qyB1+vFp59+Oui3tVgsWLx4MixahuLh40IBE3Jj
PPvsMO3bsQHV1NQ4dOoRQKIQ9e/YMahiwWq2YOXMmKioq0NbWhl27dqGjo4O9rtFocPvtt+Omm27CvHnzxmW2PJH
XwFBIJpNYs2YN3nzzTWzcuDEjO7RkyRLccccdWLFixkyRLccccdWLFixbhlck90/7Kzs5GdnT3oeYVCMSRh93TiZI7htGnTMG3atEH
P+3w+bNmyBZ988mMPPsAHH3yAkpIS3HXXXVi4cCFKskogFovR29sLv9+PVCoFKSkogEAqhUKsaV8fv98Pl8jCtjNBoRiUSgVCrR29vLCJxqtZo
FDTKZDEKhkHGdOI6D3++HUCiExWJBTk4OLBYLzGYz5HI5ZDIZ4vE4mpqa0NzcjM7OTlYqJS5HLBZDPB5HKpVivwE
NrgqFAtOnT0dJSQksFguUSiU4joPb7YbT06OjqG5PucCMRiMWw2G1MojsVi8Pl88Hq9IxInpVIpK9EMXMx
mM0wmE8xmM/R6PRwOB+rq6lBXV4fEwi1kwQ+fO/Pnzcf311+OrX/1qRvt4d3c3Vq9ejY8HDx487s2
bkJWVhaVLl+IrX/kKFi9ejEjFQqBY1GA41GwwKyWCyGWCzGHgcCAbS3t6OtrQ1er5fx5vx+P8LhMFKpFKLRKCvznKh
XFQn9RaNRuN3uYTu4TgeIP+L1euF2u+HxeOB2u5FMJk/rdlGgRZM6qVQKuVzOHhNfhoLEeDzOFrI2oQmXUqlkx1I

gEECtVkMmkyEcDiMQCKCurg4Gg4EFbaFQiPmTCYVCFgjSJIvGWwrmKKDzer3o7e1lvmfjBQoERSIRWwb+T553ND6
oVCpkZWUxnar8/HwYjUbo9XrodDpmTUJL+nNKpXLYYMftduOOO+7AmjVrABwrrT388MOYOnXquO3vWMF3pZ0gjte
V9sQTT+CBBx5AT08Pq/u3tLSgqKhoVKW0U9WVRjduqVSaceLG43F2Uad3g9DMLJFIML5NNBqF0+lEZ2cnurq6RqX
VMRBisRh6vZ4FOzTricVi7Iby7wyZTIaKigqm6ZKbmwu/34/e3l709fWhr68PgUCA3aCj0SjC4TC7MR3v90sPLgG
gv79/xO6f8cTAoHE8oNVqIZVKEQwGh9QFUiqVsNvtMJvN7JyjYJrapOPxOAtqMr8UyqVyMvLYwGrUChk2REKPOP
xOPr6+pBKpWA2m2GxWGA0GtkMngJqIuPTeWCxWGCz2SCRSBCPx+H1euFwOBAMBqHVahlvJv1mmH5TTM+kBYNBluk
xGAxQq9UQCoVwOp3o7e2FTCaDRqOB1WodkivCcRxCoRD7PWkbaZJB3B6Px4O+vj5mQBux9lEh/5Go9FBgQRlNML
h8KD3nG2gieXALNfAv+kLBWanC2KxOIOaQS39BoMB+fn5yMvLw/79+7FhwwZ2/V188cW48cYbYbPZ2H1GpefUk7
neOGs6kq77bbbb8PLLLwM4dmDz8vJw3XXX4X4eGHH0YkEkF2djauu+46tLW14fDhwxAKhZg9ezYefPBBfPjhaMu
WLXjyySfxxRdfoK+vD2VlZfjVr36vZFfjVr36Fb37zm8N+92TpSjsRBINBtLe3w+l0wuVyZZRAb0xe5XI68vDwmcT9cxwjxfjo
6OtDe3g6fz4dQKIRkMskGcepio1kNidpxHAen04lt27YhrDQWi8VYe9VUWi8VYeY90UWhJJpNwOp1s6en
pQTQaZUqzVOqzWCzsZpaeTcnKyo1cTDYfT09LA
AguQVjncME4kEa62lEqZEIskoldCNLh3hcJjNVnt6etjj9IVKMn19fbBYLJgyZQorp2q1WmYH4XQ64XA44Pf7Wdf
hmXbjogBjKNDNWaFQIC8vD3l5eTAajRllaIVCAaFQCIlEguLiYpZJownJZOi6m2icyn2kYGtgVi/9/3g8nvE4nVB
MY4RUKkUkEkFjYyProA0Gg0yQM5VKIRgMIhKJQKFQMGrAjBkzJWVBYVCAZVKBaVSyYJAmlQmk0kIhUJ2s6dtiUQ
iiMfjyMrKysjMUinsRBCNRuH3+weV2gaW5egxADY+UIbZ5/PB4/Fgy5YtKCgogN/vZzITZE8ycBnPCQ9wLJNLyQE
KIUQiEWw2GwoKClBYWIjCwkLk5eXBarVCq9XC6/Wyccrj8cBkMmHatGny8uDQCBAKpWCz+dDf38/VCoV8vLy+K6
0kXDFFVfgxRdfZC3Sd955J4LBIJ555hnk5+fjvffew2OPPPYbXX38d8Xgcr776KpYvXw6JRII77rgDAoEAX3zxBWb
OnIlbbrkF119/Pa699lrccstyMrKwjXXXDPk906WrrQTAV1MM2bMGIetOgYicc6cOXPMn43H49Dr9WOuF6e3Yh8
PPp8Pzc3N8Hq96O/vR0dHBw4dOoT+/n54vV5IJBRVh+vTpWL58OfLz81lARzNyCoRjsRicTie6uCtie6urpw8OBBvPX
WWzh69ChaWloGtSNrtVo2YCaTSmswmUyM8Dpt2jTMD8Dpt2jTMmTOHlQHGAolEgqysrAw15fFELaysrAw15fFELBaD3++Hy+VCd3c39u/
fj6qqKlaejEajyMvLQ05ODrsWnvE4bDYbrFYYr5HI5uyGGw2E0Nzf
jyJEj8Pl8rGtr9uzZmD9xMjc4eHjq1ax2iYBpyUmlsA7b2pyWzmVq5OzEeLntLg5s2Edf8DUwBmnjc+lUum4Zdj
PP//8Ub1vspKJRJJhl3LWEEOCFRNGM3+UeaRMtcUJAWDQTidTrS2tuLQoUM4cuQIKzd7PJ4RKQQpNGdNIl0dHV1jYv
N1i233ILnn38ewOnpSjsjAiOZTIacnBwAwM0334xmM4xamNnzZh1apVuPXWW9Ha2gqVSoW1a9fioosuFRES6++GKsXLk
SLpcL3//+9wEd999N9ra2tDV1QXgHlv/vz5e02114YNjE4VDh06hA8++ABL/+qz/+AY7jWElFqVRmROZyuRx
qtRpyuTwjnSSiaDX6xn3GQYjTrdyXEc+vv70djjYiKamQUql2EyGiJi7vb1gvb148CCACATZr3AqsxEV1Gi4TlUql2EyGiJ29vb148803IZF
IGA+JlkQiwVL6NMN0u93o7e7fnIz8+HwfwBAVlYWXC4tbdWdByKiCTyZhOUgkfwb3oKqasj
PFRUVMfNIi8UCrVbKjR4/iyJEj6OnpYfFpqE5fyqdK5A+mK32zFlyhRMmTIxcXFjARL+jJffPEFmpqa4HK5ES/HGd9
Eo9EwXlYoFIJKGRyFkVFRSgrK8O1464r5PhhuOTCYbsiV+IsBxHDweD1QqQqaLoaurC06n0+5a2EzF+TnL8+TXjL8WfLo
Jqzx4DATxr9Rq9ZjsryKRCHw+H/R/MUrK+vR0NDA7e77CWOTKZDPnz9pdpaSnLkHd1daG/vx9GFo5GR+PV6Pbq7u3H
o0KGMAIuy6OOt9j5j5WnBGB0UAoFArqYYbb7yRka+ivfZfZavPDC/CC/jkk08wY8YMlmxPz+Pjjh3kpDEeRlmgrK
dXzfddBMA4MILLLxz2e4YqpQFgZZMDxQlVVFX71q1+N+N/+oIVKahUs9QAUwkEmGlcjqCmPZVg+eAtDlmg
0iubmZrak83doJieTySAQCkiWA2m5GTk4VYLNLmX4OS/f8hE/Sks9aBa9xpcxOOOHdp/+kk2+Ww+OWk2G/r7++H
xeOBwOFc1yMMN3NzXjvvfdGtc4jjvf7u0hi4JQqEQQqEP+z+R4r4lL9U8Lx/UR6c
Tib6+PqYvRQGzTCaDa60nKN5mTTZmUEEV0Qz0wjLaCXU9JYPtCNbSEZ6OCXUSBFRFGUNQ1Wq3J0tjWbnz9MhR40CNkrtZLL5
AqtfrYTKZWOYqFoshRJqNsk4vIq/Se5qamnD06FN4kf4SIRC0tbXhySefRGdnJzo7O7ni5F9dwGBhOAOutHQH
b2zvoN8zKykJZWRmKiopgFHowpVsFEq1XWh9P6F6ZEtBZkZQnW1l0xRdOA4qKa6+MaosRZpFiKzbhNmYx13CalQnrC24T7hdFqVsJqrxY
09/cjFouxxgShUIbh0zIX+fxHpZEy3uVdXVY6FjNvqB6hVg9IFh6MmZYqCNbNvDqtUIlILgUbnBhZYPchwgbqtfrgzarKxvMUqqkUqUaXgkZVZWqeeRN2o0KKKQQ
0dnay34MaaA710KhECRQSCWQyGyKMKysYXxZKGJDNNiz1PAeAZrwXfP9YMNMRMkmwVW2zJxj3R5VWPfFsX9FbDJJUImmhHtqLlBTgr61eeGzRiWddSPGWZZ5
ATz/9FFVVVdi0aROOHj2K1566bhmt9RAwXEchEJhxnEe730cy3rOCI4RafwAwK5du3DVVVVdhyZIl8Pl8LP0/FHQ
6HW6++Wb89a9/HfTau++Wb89a9/HfTau+++i29+85uoqqoaVgflVJGsbsWbNhcV4CVCRCIRFpild7vEYjjFW+6ZDR/XydILkiiRx
WnU4Hi8UCUlUqVMRBStwTwL9IhBQYDn09/rFAooNFIJVKM9pWSfmXbpRKpZKptSYSCQV9qZGVlQRSQSIRKJMK
zy+VCIBBAKBSWq1GdnY2cnJYyJUFBQMGruF8dxrBwzWjIh3xcHo6e6lhTubRaBRGoxFWxqFqAxVms3nFEogEEBTUxampr
gcDjg9XoRDocDZ98FqtbIOE5otDDwVQgp4U8wbq7uweRsyUCPLS9lvIRAIOdzYeFnEy/B6vaeM2/B2gYy
eKfjiMXpQB65arWYdaKehA6L3fKCCj4+/bwS4ASA4ohkEk2+Hw+5HQrw5jMnLuFLWdZZIpI+R9ZIp6H0rJ+/4cH
Ct1gsxrLytEilUgQCAdQaKAOJMr4vnbNEkwPivBElgxalUgm7dG9Xs+I99QZJxaLoVKpYOYeDEXyYl
E4PF44PV6WUefqSiAr30u00GQ1+O47B371689NJL6OjojqjgEAgwVNVXX42rr756WP88GlNramqY2bjeOp4PNZhs
zeB
4OOvI16+++irkcjlr37z22mvDtbzNmmmvU+tPP3PXxnPb968GVdffTX++e/4pZbbhn+u89U8nU
ymYTb7Ybb7YbP50MGCmdTQIpPJGIXhMxgMoqmpCXv37s2YfqjKvpi6cuVPkpQdc+qmDUsyhjQIPEOWTlYUFBcpoR2Edpgp
+c+dO7Ny5k2mvDIRQKGT+WtSNZAYkJ2djzZZ+LL888/HtGnTnTntnGyoiZZze0NCAUAUCgEsMgMmUIBkk8nEml3RaJR1kKR
3QjmdTnR0dEAsFkMul6IgqNNklTSPJ7SNcdx6OrqQm1tLbfccGRlZWW6umakNO7hhNlyuVxZVKqQiKgLQDTgRlWF1lZWF1
FguzsbEgkkoxOHerKow679Nbp+vp6VFdXM1G4LMGsZ6Y7Jj3x0bFNMAtSjQC2jUMt7P6PSeVkO4y8QzVolnmtC8M7sBWl6kREIFBKTGD
+RHd3N+RyeUYPmoCAFqLV09Or8u9OOj8eDuro6pNGXDigEhKEgjEvNQuj0AaKhHHGUZeMw2bUUGNm+Ig08GdCxn4P2O2Gw

22O12GI1GNhkIhUJobm5GXV0dWltb4XQ64fP5YLPZkJ+fj2g0ip6eHhw9ehR79+5FT08PCgoKkJeXx0qcVLqgMYc
+09PTg2AwyGQx6FhoNBqIxWLWkMBxHEQiEfLz83HeeedhypQpGV1lXq+XyUN0dHQglUox7h9lqigw8Xg8CIfDg1r
m4/F4Rss/ZSwpi0pWFHTdpb+PyuRnaxbndILKuSUlJaisrEReXh4754lgrlQqmeYTjSmUraHuSbFYnDHhpsfxeBx
utxsul4sFof39/WhpaUFvby/LxpKO16uvvorVq1cDOHbfKS8vRzKZzOiypnNzqPNh8eLF+Oijj3jy9XDYunUrksk
kgsEgu+iLi4uhUqlQVlaGrVu34u9//ztWrlyZ0ZX2ne98Bz6fD1OmTAFwrF56PPPkNzczNmzZZo1YlAEnLn
ka4lEApvNBpvNdkKfJ0IulWZCoRBisRhSqRRLyQNgnRsk7peVlcVKL319fXC5XKy8olQqUVNTw7RUIpFIxkLrooU
IgH19fZBIJEzR2mazsRKPTCaDz+dDb28vWlpa2I093Q5jvCGRSDB16laEBwWDwuGU+4Ji4JJU
9yCi4t7d31G28pCGU3nlHWSYqA9XW1qKrq4tJBJA5MXXE0CIWi9lrtJSUlODyyy9HWVkZiouLB52LI3GMqEMx/X+
O486o1t7h9k+j0YzIz9BqtTjnnHPGrKY/1m1Lz96mI5VKZWRtj7eek+GJUadZegfZCO+jLAyVB4Fj50Vvby/LvKZ
PTOLxOAuCaREIBKy7irI2tFCwmEgkBkkbhMNhXrppcjXXHrppcjNzWXXSHpnXPPyYQxlu0nZLb8igDIbVamuT1OlI
GRyAQIBKJZJRzQ6EQmyh0d3ejpaWFleE1Gg2MRiZmuTeBYl6rX64XT6YTb7Wada5Rhq62tZVp
EkwUcx6GmpmZMn6FxH+DJ18PCbDbwjIEDg7rSbrnlFvz5z3z3zo7O1nGqa+vj7FFF00ESEqABttaIMCUy0frbwcc6xShpoN0JBIJZpcyFEQi
z9OhRfPrpp5g7dy473zo7O1nGqa+vj2U7BkIoFLLyF00ESEqABttAIMCUy0frbwcc6xShpoN0JBIJZpcyFEQiUQb
JmbRNVCoVXn/9dYhEIlbSo8GcygEU6ALHBiSdTofPkJmqlpcXIzzzjsPM2fOHHSjp+/hkYmRDvDdMTvx8Ymz9ud/Dwk
bDWWkB0IkEmXc7EcDnU43Nu3Izu7m6YTCbMMmTHsVgM27dMIZGrYpQJoO9UOGfJmK4UYPx8ygtP1jJBpqlFfDBB33nn
nccnHtH+XXXYZYrEYIzrX1tbi8OHDcDqdGfIHWi5lSgBsBIlQG7u7vhcDhYOmDmQFiEWi5mlDMdxCxUKVvVKCo
qQk5ODlM2r62txdGjRxEkhVjGKRaLMZcAi8XCqAfUmUMWTBOoET08OoETTWXXHFFRNzUEaJMyIwInl4YHBXGmGoMlF6lXD
37t345JNP8JOf/ASNjY3o7++++Hw+FgJY/TiaamJjz33HOndRaLMZcAi8XCqAfUmUMWTBOoET08OoETTWXXHFFRNzUEaJMyIwInl4YHBXGmGoMlF6lXD
37t345JNP8JOf/ASNjY3o7++Hw+FgJY/TiaamJjz33HOndRuGAnXAEeeIVKnJVoIuLgooqGxHLdbUgSCRSFj7vM1
my6jhp9fpB/5P7e7UAUTilU6nk5E5SWPEbrejvWPyFEbrejvLwc5eXlmDJlynBpYtrplIpJBKJUVtOCAQC5OT
kwGg0wufzDTsYu1wu1NXXVsZmeUChks+P8/PwhbyLpQUIikYDL8/0pBYkEgZDwCRiXeWkBpEYpA4ij8eZ5trlR
PPB5ngaLD4UBnZyfq6+tZ5i0UCg1axkUimi0SjqR6vR2trKeBv0vlmvkCTDRBBv7C
EKKaFPODOOeccWCyWDJucgX9JzG4kET7KRjidTnbM29rasHnzzZrhcLhZEZ2VlobS0lGUJaenu7kZWVhbjRBBBv7C
wkGXKQqEQBAIBu054nN2IRCL461//ikceQejj2fY90kkElx88cWYO3cuXnvtNXR0dOBHP/ORHnvsMdx+++14
ut6OwsBDTp08flH2lsrtWq0V5efkgkvfJUAw7fSrT7sgmZGIjSl5XNTN6zOkuV5PHJOczEKgxjPYgDX4SUvMzVwS9vfM+gU+b
33nu477778NJLLyEyUCuHxxx9HYDtVmxkR1UxCruRuhs9R1Kw8Kn+LyOfdmUGn2W1gPX+mOaYsJJaNwvqGvTxvsePxNmgOqKVW8MRpVzhdv+gOWndOHdBM
LI+sL0ushsh6RDiORSAbZOb0ziCw+8vLyUFlZiWnTpiEnJ4JeRQrLp1o4bzTHJ73OPFAAjfgSJ/Kdw30uKysL555
77nE/PxJIS2osUv0mk2nY13Jzc4d8PpVKobOzE319rfRkzyGg0itWrVzN+S05Ot8b5ZTFZISnQGmstb78aRWJpxV9qVJ6qQbvVO2KsYqq6iQdnsrV
paUFzczNaWlqYrIDT6cThbbTHH3886Pupd/08HpFIpSbdDYb8/gj5ObmS0SgKmKBLTVSmMmZS9
DBHmdTpeRASb7EZfLhZqaGnR2djIRQ5rs7d69G/78+xtPwpZzHhbufBRRdm4anN1w6BPoNRWnQo2p1PuUmLzMxnZg8sCuRylokBUKlIsqKO9vb
2ss6/7OxsxHoGLUCFuHmTagzDMhJhVzEjDytEpKJzjWdK8rk/GVBhtJi5rF+4XiHdm1a/41Khmb4/kBrZEjGqPn1pR7
8LeKWEaF8JCQSCTQ2NrKJ80jrTP87HtsStUWhcEbUHWNH2RMkzK+u8nF+9lMzrv4ix7CUeRPhk6GX2vbpG7kDnjQx1JRMKwbLnQ4TwbTPsQI9OpQ0l3W
TgPR28VgshubmZrS1tTEeSTqFZzIOA0pihwTbFMF29zFW7LHOXfBsZqABHf7DhBkCCakMqC9GeSKbDQxkWKb
NdFsmk2/a2Yp0c+10ZXPqEC3N5CeJHUjMCfZYWUkMnpPu4SSQSVn0gSYzm5mZ0dXWNKHEik8SlVnz9JnpHO5/Ps9nMSkoSiSR
jUpr+l6yXotEoamr4XA4mAUdHdeeli93SxIMmullZWdDpdNBRoNKzKzZgHzkKJtMgTGtFwDTyyM5CJ/Px7TnKHOeTCY
ZgZ9U+6kZiibj1ODADHOPjiUQi1plnMBiMMjYKD7hNYWiBXYy8PGJd/vJldcRB8DziqmLmwJ3Ot5onzYDE+Fh
33opnn30WwDEX+yeeeKAKHDx+GRCLBnXfeycbfvrL/Wku7IG8/QPGvo8cx6kMHDx5kmGMcf4YGWXmHT02oO+TSdCK2
kC2gn43FMJBJ49913sX/pOf/vc0MLixsJkJ11dmzZoF0O+Bk/FJKvkrRZfcFvWqsLzGfHTW4G+yIiglmqEnxLFKhSFLdvRjcPYfWSTSMbdfY6F5j
MaDRizpw5OP/883HHugTgWLatvb0dWLAt/vb0f2V6/eqsbbzGbKnR4MWwQdmdrhMK3QKfDAjBBVdUjk2bA7X7HZxVBDdLzbebebzgLxjZX6AWwKuoSTK7bC+
Z+vv7GbGWbj7pn7mw1xB5FWb+9Y4p7/
ZwFik2mw333vXcf21eXywwXy4VIJIL+/n5UVVWx9Wi1WpiqcF2b2YLG2YZVk1tSZE0uUgQoK0RjUQ0ZjOT3SQqY6
vr4+VboFjwYVOp2N2QnK5HCKRCRA6HA+3t7t7dRAFGuu1G+pJIJFj3JWWQ0rNkNRkU4h7DdtPNOhUCgyuhYH4ubn
f+xpeeeUVvittJCxatAjPbPPPM67aiH4m60mh2lm4d8uGHH+IHP/gBDh8+DOCYwesWuw9oiCYoewfvfr/8pLDMPfxDX13
/fMTvPVVdadFoFLt27cVjOUaTGzBqasvqgCXVVxduFqLDHhd0ro+dxZuY/bl7UlZDVWRx4xTcUFeFgW/HcSJZ2vbYwmTV
vvIP169fD6XQOu57fsMXWWqXEHsA4vnrNnFmHvNJSRLAaTdbMmMaEEFCqMe0BopgM2BxzEWi+GFF17A7373u3nBVgXqnrVZ7cQsPU2K3oEF7qcUAbdtqfC2UmTnL
Tpk2j/v4PP/wQLwEZfGVlJa655ub655uNTG08HWalePC5ygHgZ9Qc//lcJ5J5TPRuAEFQDzP45+e5bf+Pq8sAV
44QaokWq88soreOWVV5BIJKBQKLBw4Ew4UI8+OC2IlqGJ42MH+/n+DmwJmCJRwdz9XqsdLnVfY6z9Jm3tuIhGajDsyHr0zcqmXwxWtnXXp7h7WthsKoVvPTK5C3xhzNSQ/w5
dixYtmlAZjjxMBSS6Q8CcFTSRPQp17HR0dWLZsGbRabYaZK5nppfeUsaS5DvSm1VaW1uxa9cu1jRUFCA888/H2V
lZRnyFAMX2g7is6V3hdL3kucbcR0H8i2VSiUKCgpQXFwMs9kMoVCI17du345NPPsHevXtZUwl
Ky8tht9shFApZoEmfNxgMLIOarvrVaAIrvvZVJEJZUzSzJNKpUz2xGw2Mz80amRwPXRnkmQw+/3o7Oy/lBXhhYYSH8fj/
27NmD1tZW+P+P1+xGIx6x6GYjHG06nS6dBJuW1anVDIP0GpT0f9NBAYyONJ+MXktHgXNIvxhUz2H33xDM1HGC+4fWpyxmcz1CF+1wNZgDFPwDMkkNoKoQ0WtCiHmlf6UgBtjR1aJdpyepxumuCPQ
B7fjoo4+wZMmSGr5Kq8yWk7EhkEwRLjmxR1Y4q4hwLnnjiCTzwwAMn9FhugsuvTW2vjZxiw1dZJ1YGqHqvrToUtkojH
UPqZSKezcuRPvvv/6Q8+tm3bhqqqOowspEgkEgkknrpfBvCeZTIbLL78c2c157LRYtWjRsl04

8Hkc0GmUt/BO9j6ca8Xgc77//Pn7xi18wErjZbMa3vvUtZGdns0mDUqmEyWSCxWJBIBBAXV0dyx6p1Wqce+65mD5
9OmKxGI4cOYJ9+/ahqqoK+/btw/nnn4/i4mI28aCSBVnddHR0YNeuXdixYweam5vZtikUCtxxxx24+uqrMW/ePDi
dTrS3t6O0tBRFRUUZGcCuri5UV1dDLBYjmUzilVdewVtvvcWOO1knEMfHYrFg6dKluOWWW0bdGDDc70fHMJlMYv/
+/dizZw/TWNq5cyd7r1Qqxdy5c5c5kXIHGezj33XCxbtuyktmMiMdbzNJFIYM2aNVi7di127NiB2tpaxqOjjk6FQoE
//OEPuOeee05pF+BQGGr/0rNzPp+PeUVu2LABq1evzlBrp7K1VuF3W7Hf//3f+OCCy44XbszJMZ7rBnL/fuMDow
A4L777sPf/vY3zJgxAzqdDn/5y1/w6quv4vHHH8eCBQtw+PBhJuT4X//1X1X3j22WfhcDig0WjwyiuvICcnZ1jtkVN
VStu7dy+uvfZalJaW4mtf+xpmzpwJnU6HVCqFuro61NTUsE4h8ePE9AyGQyLFq0CFdffTWWLVuGwsLCUZVjEok
EQqEQAoEAS4V3dnaivb0dHMfh8ssvx4IFCwadoKlUCj09PUygrqGhgd2E/H4/1Go1LrnkEkilUnz7298e9UC
1ZcsWvPfee6x7kGYuZDODBQsWaP/sccJq1evxNPPIGuri64XC7MnDkT3/jGNyASibBkyRJ0d3fjvff
ew3vvvTcoU2Gz2XDddddhxYoVOO+884ac2Qe0EezatQs7d+7EoUOHsHfv3kGdYEajEajETqdjnEsqGOqsbGR3Vyzs7P
x7W9/G9/73vdQXFx80vt9ukppHMdh69atePnll/HRRx+xbh+LxYK/f//znuP3228fFX+lE9s/pdOKzzzz7DEOk08+OaK
BZm5uLoqKipBMJtHV1TVsluviy/GD3/4Q1xxzrTXXjk/Hw4cpprsHTpUixZsuS
U+deNBiPtYzgcxo4dO/DFF1+wBpW1a9cOKWkBHMu03HHHHfjD3+IgoKCU7H5x8x8VYz9NoNIp33nkHzz
osUEsFuM3v/kNfvKTn5z6oI8w3mPNWEppAAvvPACHnzwwdoj+HAh7F79+5BpRmlUomcncxBxGoGhAZ2dncf9PqVSK
+xYTJBmK4n+BUka/H4iCfDmo5r6urw+HDh7F79+5BpRmlUomcncxBxGoFOr1dDr9cwXLBgMoqGhAZ2dncf9PqVSiZK
SEuTm5sLn86G1G1tZWRBEeL2bNnY8WKFZg6dkUqkUikaGxvx+eefo6eefo6WlJYNrMRX3nAoKirC/PnzUVxcjJ
phoYGNDY2QiaT4YorrrsCcOXNO+OJPjdpz5PBpN443Y+eOeffz5mz56N8nBHtM6Jk5OTkeKaxgO8FPz57pcxnNwGWvHFDRs2oKmptkPM+8tVWWFvWSFh8z/zxHNT6upwRmrz3thpM
OHEB9ff0gHaaRIBUCiEzWOzqvjphiMRiIOTA0iHUCiEzWZjdjjl5eSRyw/5xwCLz3uB/uWh7aqmEaSGMAGNsVKvHyy3nn
EjuOwf/9+bNu2DQcOAOHEBvby/kcjmRiQEzWZjdjjk5ewTU9wlAqCcjl5eaSyw/5/xqmmzBHkmqaEZaMAKFi/9ILP2//L/Hfy
A82sy2FQQUqkUqqursX79euzXeuzevXtIXpZWq8XmKQBXXmhcqI6axebqQssOg
Nzz/+8Y+h0+l08ZOAM4q8vVoMRJJ+/nnn8cvfvELrF+/HpdddhkAoLe3FxZvKzzVzx2xdOtnpXKzNcRyOHj2gx2j
XY/Xq1dizZ8+YghbhBHFdQn2GBtsViYFUJeXh5Ph/WrFkzpHEmfYZKGEqlkpmR2u12ZGWEtjbsyvwZ8sInmTanclpmAGWm4BZeUqFv
cjEYjrrvuOsyePRuJRIJ1VxAzAtKWBRs3bhxySxOR+yorkR8bk
lUmHdunV47bXXX0NjYyOT3r7qwbK//ag/fW47LLLxv3mTd5rwWAwg6tgtVpRzUBEvVYRvBPb9u9ffBrXEyfVyXHTRRRdDg
yZAkuvfRSlJWVZRCRCIB8Lr9WLDhg3g3YsmULgMJs7MXPmTmGi12gxMvWFi655JGRcGW0goKzNl2LLuU9anYKrWIqx
lYppVJpMJOp0O0LpcLnZ2dGdM77M98+xjW/goosuOi3ZJ4VBy+X64pn6rBm3HNNdBbxXtSuHMjeBB4fO6YWpuOmgPg5mqD9u
3tHDGxmMfk8kPv/8cMfz8/8T6ptYtfq2GYMJSEThevr7dFy0/AhRXYVxcj4zm+L6ZotIGWPXBpANwmgmU7Ir9vB4Pd
suPTSS5mu2vvTp03HNNddMisC4cnYkadSM3BWVxrr3Dsc4GOCPtz4/NpaaepaTGkVmgCmo1/4NUT39ZZjXtLb2nWeXScVGRZwRSemTzHxXgwATsuWGEvN+OyH1ZJUN9WpAebGE67WMzbc3I4Gfy
AnEgk4goLC7kHHniACwQC3Asvb27m+vj7bAB4Nrb27m+vj7ObrdzALjs7Gzu2WefWefHfV39Pf3cwC4/v7+cd/+WCzGrVq1iov+WCzGrVq1iov
FYuO2zmg0yh08eJD76KOPkKOPuAbNnBbtmzh3n//mHn//mhn//fe7pp5/mHn/8ce6hhHnnkEW7t2rVcv1cXFw6HuVQNeS6Eok
Et3fvXu7FF1/kHnzwQe53/sdt3btWq65uXlXlU2xyLxbi//e1v3D333MN97eEn9WwnMO6tdbI0SuHvdqulnRQfjjrBQKkxfjjyDdIV3wapgvc4Y
N3Pbt27lDhw6Nar19fX3cNdfdc0UF1J0xaL2bq1Uil33nnnnncffw3/3PPPcw888ACn0WjY65/dddhm3efN
mLplMDlqvx+PhPvkE+6Pf/wjdfd88NIk8+DfL/wjd88wd9CXXHI+xJUdt83bDzU+VPX+I0JP4L4vTy/6qqO2dyOr56pC9
Go5G77bbbbuC+++5JtfF2tLXfzzTdzIpFoMxM8MXKRSKbd8+XLtde4/vOf3ZTKZRrYc
WsVjMXXzxzvf/97rq+vb8J4P4p4m4DicbJmIfHQ4H99prr3G33HxHo3G33XZbuB31xzm2aHY9/OXXX3
11YO2QaPRcPfccqbPfw9XVVU16azWyu4aWwWIujuGhQ4ad0Omsd/rjjvu43+7RMKwp4xEAeI/37wDuO9
WW3H//fejvLwcwGZduxF/+8hfEYjGsHY3sbvKwT/9eZcS7MLEX4XH0gwXYcbyD2ZKMzVZUVyTvXhv8W9KsRZ8Fo7HjeWq0WQKLlnw
FBYNmgF6vF48//zLv4+uo3CGebhi8ZWKy8W4U4NMC1ldaB3i9sGkRS4fuOcrFK4xaeQsCYRsDrRZZcgK1eJD4I3IZhMJlrt3V/laI/dlnhH
qq6+eFMeQhD+HsjCpqanBhg0bsGIXNwkHXkLrI7PrcuXlnQlUusFsNq1tZVl+SoqKnDzcHqtQGD0wgOMOt
1bB1yuRwmk2mQBQvy+99Ag3gXq8XVJD2aRXVqsVZtZZSTrcrSrcZfg8DE7yPHcaitrwcWDRvw8cc
fY/369ax0ax0X3Ed/3XvrjKzcvExqxkb02amWfVUhlq17aKz0tSPlxMsSddQ0atD1m7MN4+IDrtQ3IcWS/t+p5qy+jSoF7jEdpDg2+k6Eq+kbmNDl+AaoTl76NTm8a+I
2OytPGo1GyGQy5i1UPDCPYk1SEeu8nq9hhHPqX2WTC3JWNdsNjNBAxMDnzp2oq6tDQ4TmeOMjSnsLZL0ygcSgsLZVE16xWK+x
hFAoxM9+srCzmKycQCHDw4EF4q2tCrnn1W7q3dHDnDs1+PbhwJ49e7e3DdddBIBAwfRsKaFwuqqrB169Y
M3prVasWVV16Jq666666CvPnzx/WC3+YTkwxw1CRAkcF+6cU/kzW3gYJxMJtHT08O1MEBAe+1+/3MM1O1X5001aTSQS6Onpwe49e/ZDZZ5+
hsbFxkE4QLbFYDLmiRm+duFwoFEkgjajEsSqWM1W6FQQlVSMdNWMnB2Op0QCoVM8uBA5OTnQ6/
IpVLYs2cPuzYwcpZYRrEYc+fOxfPxOxYZIpGKGDnfZs2QGiXmvnZWEkpEdPWxxtKYnSCT
45je/iXPPPRff+973KMk0MQ/k8kksrOz/98ggTvvvBNB/oziPwQCTTb3/7W/zoRz86pYEYEHlxiNA4Y
KjH70ox/h9ddfRf+19fH3bt2i/HmdDZsxyH9jn86G6unrE2vosGEsxmM7sBEfma+FeJRCJ
jcEsXb3O73cdVC/53hVQqRVFREREqEQwGAKBXp7e+HaZaM55rs0Ov1UKlUrHUpoHWMnUpnStHNnIQtSmpDKRiSizL
iN2ehUIhoNAqvqd14uuri6EqEQiEqhCxKpoPK6Ot7dVZk1IVFqaonjGsiEilUk0z1QzeMdKJW8xsZenJSkQQ8j8AMQsmmWZbUMq8cR4k6

SKbq7u5FMJjOMNYFjgbJWq2WCi+nBI71eUFCA/Px8FBYWwmw2w+Vyoa+vDwqFAgaDAQUFBZgxYwZsNhvTj4nH40g
mk5DJZIzkGwgEmKAk2XoAx1r8Gxsb2UIq29FodERfrrMRA/0wKYDKz8+HXq9HU1MT3G73sBMftVrNxkkKksRiMZv
QUABGNkzp2kOBQACRSATZ2dnIzc1lQpS00EQv3b+PLJzSJ7KkLK1Wq5kcgk6ng1wuH6Tknv44EAjg4MGDOO+889i
EMl1Nm/uS40bnajAYZMrbpOXEfSm2KZVKUVhYiKKiogxNvFAohHvvvRevvfYa+73mz5/PJrZqtRr5+fmw2+1s0mA
0GpmukclkGnKy7vF40NLSwiyFVCrVoP1Uq9Ww2Wx8V9rJYiTrkFdeeQXz5s3DT3/6U3zrW9/C5s2bsWjRIng8nhH
Z96eqK62mpgarVq1iwl7JZJJdhIFAAMFgkAUydCElk8kJV3/VaDTIysqCWq1mKrSRSATBYBBCoRAymYy1zwuFQjb
Do4yJVquFTqeDVCplr5HMgEQiyfg8eS+lB6IkYAmADUgDF4lEwt6XnZ3NApZ0Lya1Wo2srCxEIhFm/tne3o6Ojg7
mQE2eUOkLkYqtVisKCgpQWFgIu90+5MVON2LaR8qEBYNBJkBHCx1PgUAAq9XKMkl2ux1arRYikYiJqAUCgYxjTpm
hgaJ2YrGYzdbLyspQUFCA7OxsSCQSdHV1obu7m9kDyGQyRgoOh8Pw+XwZprROp3NI4rpYLIZGo2F+aAKBgP22FID
TbzDQqoOGGRLRO5Mw8AY8mUGdbtRZRQbJA20u4vE4AoEAotFoxusSiYR1hdINnpZkMslu/OTlplarYbFYmOdXLBa
Dy+WC0+lkN2S6psmiggQJ3W43EokEywSeCt8v8orjLVuGBmVw0zO5FESdyLooyFQoFMyA/HhZdwC49NJLcf/994/
5O0cC35U2oCvtt7/9Lbq6uvDmm28CwKgDo8nelUZZoHg8jkgkgp6eHjgcDnR1dcHhcMDlcqG/vx+hUCjDJDLdKFK
hUDBjyPS/RqNx3EpzJ7p/ZxrOln3kOA5erxctLS0Ih8PIzs5m2cdEIoENGzZg4cKFkMlko+oMI88l8p0izayWlhb
mlRaNRiEQCFBQUIDc3FymRs1xHOt4s9ls0Ov1zChZKBRCLpczA8uenh50dnaip6cHIpFoyICAysQej4dlTrVaLax
WK1QqFUKhEGpqarBo0SI2GyaByKamJoRCIRYI0qSBMgrd3d1obW1FW1sbWltb0dPTw8xXyUi3sbER1dXV6OvrY+a
wpJ4ci8WY8atGo2Gzd/L54tKMZktKSpg8BU0wqMR8vGMyWc/TQCCAbdu2YfXq1di4cSMAsG5Fsghxu93w+/1sQrF
w4UJUVlayknq6iXZHRwdWrFgBs9nMsin025Bem9PphMPhYNk/OpfSF6lUmuHHRlkgqVSK3t5edHR0MF8wWoRCYcb
klsy8B2YxqVQZCAQytj8Wiw3yVEs33ZVKpfB6vSzbQhPPdOsarVbLJrcqlYp19KZSqQxD30gkgubm5oyuveEw0Ii
YsrYEoVAIhUJx3ODHbDYjlUoxexkATP1boVDg2muvxZ/+9CfeEuRksXXrViSTSQSDQYhEIuTn56O4uBgqlQr//Oc
/UV9fj7feeivjMyaTCb/4xS+GzAoBp84S5ETXLZFIMlKfNpttQrZpPDEZrCQmGmfDPlI6fCBoJqlUKse0j+nXEaX
0iVcyVlgsljF/ZrQYjtdARPKhoFQqodPpkJubi/POO++430EljImQGxgLJtt5qtfrcfXVV+Pqq68+6XXRcaysrBx
yHykoraioOOnvmghwxzH+Hm/+DfeltlEsFsvwtaNFIBBAr9dDoVAM2q6uri58+OGH+Nvf/sbEI2fMmIHnn38ecrk
cDocDfr+fTRIKCwtZ6ZJU+qkiMXAfgdNjCXLWBEbAscF8YFdaMBjE7bffjurqaixduhQ5OTn44IMP8Nxzz+HHP/
x7r333tO92TzOAHBfGr7W1NQwZ+1UKoXs7GxYp09KZSqQxD30gkgubm5oyuveEw0IiYsrYEoVAIhUJx3ODHbDYjlUoxexkATP1boVDg2muvxZ/+9CfeEuRksXXrViSTSQSDQYhEIuTn56O4uBgqlQr//Oc/UV9fj7feeivjMyaTCb/4xS+GzAoBp84S5ETXLZFIMlKfNpttQrZpPDEZrCQmGmfDPlI6fCBoJqlUKse0j+nXEaX
PFggEMBgMJ/RZm82Gu+++G9/97nfx6quv6quv4sEHH0R1dTWuvPJKxiYxpfH48Gll146KcbMsyo
wIv8XALj55puxadMmrFq1KoOQDRwTinvuueewa9euAs88+i6amJrz00kuQSCQsdatWq6FSqVhX2sD
SAP1N91Mix+6xgDygBqaRyd27v78fwWAQKpUKWq2WlURoEQqFsFqtsFgsiEajGZ8lwcOGhgb09fVBJBJlzEZoBj3
w/0AgAJ/PB6lUipycHFgslgyypFwuR39/P5xOJxobG3HkyBG0tLQwXhYRJul/qVSK4uJiFBcXoJiFBcXo6SkhMn7p3fAEDe
IHgcCATQ2NqKurm5UtfGBMBgMqKiowLRp09hSUFAApVIJoVAIt9ud0dlFZF3qnDEYDOz3UCgU0Gg0sqfqtKCsrg9l
sHjSAdnd3Y/PmzaitrUV3dzfi8TjrxMvKyoJEIoHH40F0Oh2hHgfHcYhGo6wlngim27Ztw7Zt28ZzNwAcK+EQ8ZX4KSN5Wx0PJpMJeXl5LMCljpGBgWYwGMSBAwcgEonQgEonQ09PDSNv9/f2sK4U4Muk
cCVLJpv+Li4sxderUNboNzGYzYrEYnE4n+Pz+/+2sz24Bj16hEImGcH41GwwjEXq83I8D3er2IRCJ
QKpWsBJ1IJNDb24v29nb4fD4kEgnU1dXh97//Pz+VSg2SLUiXHAiFQvB4PHjooYcGnNzGYzYrEYnE4n8+Pz+/1s24Bj16hEImGcH41GwwjEXq83I8D3er2IRCJ
PCRHaB6qhn4lQqVTQaDQwGAwoLCxEbm4u9Ho95HI5du7ciT//+c9wOp3o6+uDz+dj4zaVigd2pUkkEuj1emYrJBA
u7m7W2TYctFot6xB9++23UVdXh8WLFwPAuN1rx7Kes4p8PVxX2kBu0fbt23HppZfin//8J6YtWzbsOk9VV1p7ezt
27NjBsj8cxw1qzxwY2BDxmi66WCwGn88Hr9c7Jg+tgRCLxcyZnBaVSsVubLRNANhFQ11UbrcbXq8XuQk0w+Gcup/
SO0FoMKD1DHxOIBCwG2gikcggVZLWD22vVqtlN3WbzQalUsm0ZdJvxpFIhJEtHQ4H+vr6WHddeodc+sxZKpXCbDY
zEu5YCOmhUIi5q7e3t7OOuPRaPs1+02fYGo0GQqGQqGQZeno94nFYgiFQkxaYahLlzzRSkpKYDAYIBAI0NbWjrwEok
EVCoV9Ho9xGIxu9H29PScwNly8iDuDsdxjHB9sqDzcbygUqnAcRzz7pNIJBlt2BqNhmn7kPZWujcakb4nuhsvPaN
YWlrKgsehbtTUsJFOAKamlXRycLoVT/pjWihLRqRv6jxLz7rGYrFBWbJ02RBaFx0zsVjM2s8pYBk4Hp4lty0ew6C
yshKPPfbYuK7z37orjUo8wLGbRF5eHm644QY8/dHH+OUvfzniOid7V9pQIGNZ0tYJBAI
ZXWkDF+r4Ic2OdDL3eGK8O2GoPV4qlZ7SenwgEEBbWxvTR8nOzmZlPo7jxrSPxyNYjoRIJIKOjg709vbC7XazIJJ
80saKYDCIuro6pt1Df0OhUEYJQyqVYs+ePSyIp4617OxsZGVlwe12M5FNas0eqFFCOiw1NTVDlifVajUT/QSOBU9
UItZoNKybiILaeDwOn883SGIgvV2Y/spkMjb7phk6BdV6vR4CgQDt7e1YtmwzysrKUFxczH5PCixHI0xJmSitVgu
lUsl0ZWpra7Fnzx44HA0qKCCRSODz+dHdDT05NBRh0YqKhUKqbDRN1a9He05fPJ2pVGQRt1Lw4HCqLTO756e3v
R2tqKzs5OliGMRCJYvHgxCgsLYTKZoNvqWZDa19fHziOn08k6q2KxGJuAUTCpVCozqA3p1wJ1Acfj8YyyyJWVb0ie
PtO203oEUgnRpBOq4VCgUGQs9J5FIUF1djXPOOQepVIqUUlugyZANDlVKBRMbkGhUDA6BO0rTWopuE4P5mkbqct
NIBAgHo+zjDXpLolEIkYN6OnpydDGCwQC0Gg00Ol0Ol0KCoqQmlpKWKxGNra2uEvLy8GNra2uEz+Qb9dnPnzsU3vvENittPLBo0SJ

oNBq43W783//9HwBkkLB/+MMfYvHixfjud7973KAImPxdacMhJyeHzRYnG8bztxt4I+A4jpUX0jNMwWAQkUgEMpk
MhYWFKCgoQF5eXsbnKSPg9/uZKnd7eztrKT9y5AiOHDkybKlKIBDAYrGwgYTawA0GA0pKSlBRUcEWcutOt0Zpa2t
DS0sL/H4/U76mGzoNsJQZI0VhWt9A9PT0oKmpCQ6HA4lEAnPnzkVhYeEgbRLabgDQ6XSYN2/ecX/z8eyGSSaTqKm
pYe31CoWCZbooA5muuTSabfP7/SyYo8zbWDDS/o1lfyUSCfLz8zOeMxqNmD9/PubPnz+mbZooTLautLFsi1QqHdF
S5HjnaUlJyQlt42RBPB6HUCjElVdeOamO4XiC70obJ5DQXyqVQl5eHoB/kbDfeecdvPTSS4hEInjssccy0nSbNm0
6LV4whKamJmzatAm7du3C4cOHIRKJWJRPfIh04uTAWYlIJIJEImGq1yeiPxSNRjP8qmhJ/7+/vx96vR7Z2dlMd4V
mswDYDJjS6bSQ6F9zczPeeOONYflO6Uu6OjdZBphMJjY7lslkEIvFcLvdTA34yJEjo1ZoFggEyMnJgUgkQiwWQ39
/f0Z2cCRotVpkZ2dDrVYzjlEymYTD4ch4n9frRWtrK/bt2ze2gzEKKJVK5OXlITc3l9l5RCIRVFVVobGxcdD7iZh
PWZ22tjZEo1EWbFVUVGDGjBmYMWMGpk+fjtLSUjYp4DiOcSuGUnym8l4gEIDFYmElvONBJBIxQvpQr42VDC6RSBi
3bjwxMIgEjmWEiCMkEokgl8sZP2gyoLe3F83NzRAKhYw3o9frmXAoNUWcrTdVHjxOBmdVYDQcyBvqhhtuwC9+8Qv
2/FNPPYV//OMfmDp16pCfO1Xka8pqjQdEIhFyc3ORl5fHSIWUpk8noiaTSQiFQpbynwjLh9MFStnSolarGRmRBPg
ikciQxFqBQMD0oex2O/Lz8zFlyhSWnZk6deogPZtkMone3l60tbVh27ZtWLx4MWQyGVOUrq+vZ35oNTU1g3g0crk
ceXl5KCgogE6ng0QiYXwVIsUnk0n09/ejr6+PedfV1taitrZ2yH3Iz8+HxWJBMpnEwYMHWXA7EB6PBx6PB/X19fj
www8zXsvJyYFCoUB3d/egYMhisbAOrYElLOJ0pYvIkW0HHROtVouSkhLMmTMHubm5zNKirq4Ozc3NUCgUjAyarsp
N4n5Wq5XZhZCoHKmfG41GRmAd65JIJHD48GG8/vrr6OrqQnt7Ozo7O1lnpEQiGdY2RygUMkJ5RUUFCgsL2TETi8X
M5qCiogIWiwUulws+n4+drwaDAVlZWRAIBCzzld4JJhQKEQwG0dHRga6uLnR0dMDj8bCOVSqhb926FVVVVcfl4dx
yyy2MJD9c99lwzyUSiUHq5uS1ZjAYkJOTwxSPqSTIcRwjS6eXpIijld4oQmMsEdD1ej1ThR8NaHw+FWra6aAGjPQ
yHzUuCASCDOsOsVgMj8cDt9uNWCyWwQEbaEOSTvzmOA4KhQIulwtbtmxhvxt1e5lMJgiFQni9XvZ+asc3m83MpYC
0iYiATWKpHo+HCUSOVJ7lOI5ZSlHzBX0/nVdDdQ7G43G0tLRAqVSy63gojPcx/LcmX+t0ukEk7IULFw5pcQAcC0o
WLFgw5Guninx95MgRvPvuu2zgEQgELNOS7r0zsA6dXv+lE/pkSKdCoZDZeRAROP2xUqlEMBhEf38/gH+poIrFYqR
SKda1QLPU9IVUtwfWkkfKgqXL99O6fT4fAoEA4vE4UqkUVCoVDAYDsrOzkZ+fz25eI4HjOHbDIk0ZpVLJLuaJ5iy
RqBnxVcYqrRCNRjPa/L1eL+tqsdvtKC8vZ55bwDHeREdHB7vZqlQq1rlCkwYihlPgONT1QgHOUHYKxFc7EUmDMxU
CgYARsyl7erKgm8RQRG3y1hotKEAkG5l07uWZOuxTAJAuZTLU/2RBQV17pJhNYyl1TxFfJx6PIxgMMj+9dI4QceX
Su8NCoRAikQhr1KAOX1KWPhMx1PklkUjYGEX7Tt1tsVjsuNYqJHNBnnMUSNF9ijpK6bem5dxzz8Xtt98+rvv3b0m
+JoxkDUKaRfv27cPFF1+MP/3pT/jOd74z7LrONPI1lXOoA6qvr491haQTrNNJiOTibjKZoNPpJkwXZLISPscTZ8s
+chzHeE+RSCSjpTmRSOCDDz5AYWEhM2Q1mUxQq9WsnEe+bOkKuuldk0T2PHz4MPbt28e4RGq1GlOmTEFJSQkL9OV
yObKzs2GxWJCdnQ2lUskIrwKBADKZDKlUimmkOJ1ONuNNb4Ee7QIATqcTF154IQoLC5ktSCwWg8PhQCwWYy3R6bY
5HMehp6cHR48eZUtbWxsMBgNMJhMrYbW2tuLo0aPo6+tjhOBAIMCygKOBWq1m3npGo5GVs3U6HcxmM6ZPn45ly5b
BarVmHFO6qQkEAqxbtw5z585FKBTKKHuTXyH9Heq5SCTC7ISGIgWTThYR4ql7TSAQIBgMZmRABj4mUnz6pIq63M4
0EJGZAjbqNqYAK5FIsBInaTjRMtCGJD3TBhzzjjx06BDTjaPWfK/Xi97eXhb8qdVq5k9IVAnaNtK/crvdGcHcWAN
nukbJub63t3dQhnmo3yYWiw07kb/pppvw97//nSdfnyxuu+02vPzyywDAUoRXXnklfv/730n/HHP/4xtm7diur
qagDAPffcM2JQBJx55GuJRILCwkIUFhaO70aNIyYb4XMicDbsI5WrBoKMY+fNmzfkPlIr+5mKkUi7w5XcCbm5ucj
NzR1RAoQwVEdiJBJhZVYqbVAmghatVnvCkzLKTFKLvt1un3Tn6XCdmrFYjDVU0DLS/z6fD6lUCjNnzkRubi5Rubi5sNht
MJhNUKhUEAgE6OzvR0dHBAkOpVMpKfv4XMicDbsI5WrBoKMY+fNmzfkPlIr+5mKkUi7w5XcCbm5ucjvauY
fSoK/Pp8vwyOQvA5lMhnjWg48ZuShRly3vr4+FtwVFRXBbrcjmUyio6MDbrc7Q++JxG1pv3jy9UniiiuuQFZWFrx
eL2699VbceeEJGZAjbqNqYAK5FIsBInaTjRMtCGJD3TBhzzjjx06BDTjaPWfK/Xi97eXhb8qdVq5k9IVAnaNtK/crvdGcHcWAN
qagDAPffcM2JQBJx55GuJRILCwkIUFhaO70aNIyYb4XMicDbsI5WrBoKMY+fNmzfkPlIr+5mKkUi7w5XcCbm5ucj
5eTmsVivbJ1pIvJG6g2hikkwm2SRpoBQG8X6oUSMcDqOqqgpHjhzJ2EehUAiz2QyRSMT4XzabDRqNJmN98XicaUf
R96VnFOi8CwQC6O7uzhAEPZkxRCgUQq/XQ6PRwOPxAYPqVQKo9GIvLw81iWpUChgNBoZ18tsNsNisNisSAvL49JRQz
saCKLDerGHY1I52QA8ZL8fj8SiQQ7h+PxOBOZTW+0OB2IRqPo6upCb28vuxaJzC4Wi1mQ7HA40Nraiurqahw6dAg
qlQrl5eWw2+1WWz2+1QKBQiUSsfFdaWoqlS5cC4LvSTgqjtQT59a9/jVWrVmH//v3Hxeep6kpramrC5s2bWfcWgIxMDbW
6ppNZBz4eaAJLfjdjAenAUDZm4AKAiSBSh9OJgrrUKJOUTv6jwVCj0bCav8/nYzflgftFthI2m42RUq1WK+syEwq
FbIAhkie10UokEqYXpVAoWBdKPB5nz+fk5Jzw4BOJRNDY2Ij6nq0t7ezmjpxG6irL73Dj7r8iKRJGG1wSe3/2dn
ZkEql6OrqYjeuWCwGpVIJo9EImUzGvJEoWD7VEIlE0Ol0lLAAa6zk71PrS7VzSAwmyYaFjDRzLTPj9/jO2k2gk6PV

6xONxhMPhcbVIGQkjkWoFAgHLhFMGnDJI4wXyDSP/wHR/v4FjDGVQaXuSySQEAgEUCgVkMhkLNgcS0ekcHS6LT4H
3cK+TSTipUafbPFFmniZyoVDouMdOoVAw/0CDwcCyyun3kGQymWHfRNl6nU4HjUbD7js9PT3o6upivxepX9P1GQq
FGJ8qFovB7XZPyNhxySWX4Cc/+cm4rvPfsittNN1owNgCozOtKy0d1JLe09MDr9cL4F/txZRhoIuOniNLhLFsM3V
XkM9Vf38/a91O9xwTiUT44osvcMUVVzACZDoo3Z5MJpmWy3D7RUFFPB5nJMjJgInuSqPglzI49Njv97MBuaSkBNO
mTRuW+DkcudXn86GpqYkt1L4fDAYzsgVyuRx79+5ldixFRUUoKipCYWEhrFYr/H4/PB4PlEolC7JJ76S7u5sFae3
t7di7dy+6uroytsNgMGDGjBkQiURsxknt1yqVCqlUCg0NDWhqamKt1BGBlpaVrpBlpaWoqCggBGqqVMPAOx2O3J
zc1kATiKhPT097HdraWmBy+ViwpoymQwcx8HhcKC2tpaVnNJtG4LBIFwuF1M+p/IcrT8SiWR4+6VrC6Vfa6Wlpbj
kkkuY3lYqlUJvby/7rQUCAbZs2YLS0lJmepweTEaj3w+XxQKpWw2WzIycmB1WplavsKhYIdb5q
hUxmbTFkHguQnXC4X/H4/a6Wna506JHt6eliwMNACw+FwzNAjzkYQXYEWCqImA0jglDJFEomdbFS5thiscBut6O
yshIzZsxAOBxmyvdkQktZ8Isuugh33nk35V2sti6dSuSySCwSBEIhHy8/NRXFwMlUoFl8uFb37zmh48CCcTid
EIhF+8IMf4LHHHhvxBzrTutXrYbqbXrYbqbRqgaPlg9G8gV0o6GOi7Gm0J1OJw4cOMCWlpY
WpgCdnZ0NNm80Nm80Gi8WClpYW1NXVQSKRsJvijBkzxzxqiRSaTDRKYHA8YjUYjUacd955I75vPEmtHMehvb0dPT0
9zL/JbreP+fhUYCxTQHvSYjEpDhjQRz1jmQqZZnBCmjStw8ibUqCbkCiamVdB+cKmphppRu+gUHc74Qn
E2Hbd9vc9l0mFfUMYIYAeOAgj+v7mhLGPkJxBuHHfdaA06vNBOm44yfR1HYfXwRwFv4F1X5t9fA+f8/1ru6uQfFYBx+Y7ZN+KdZ3ZKRLHcHzPQ/KbNlFW0qrK
rCQnbpI5wDnrI/Pn2Rgeq0P01XXNC2xRegt3zSn/Ut9aG9arFb6YKqvpCBbUoeTBp6U8FR+WwnN6LzG1VK+88yvxFIo/A3P8q1U1ZaxFrvS4+5CmVz7TH3P
8l6DtUjXG6DmM3nrAfNBEg0eh9Lpa7FdOfNVGbDS4/LJNI3R9VvT2bsGDY0yehvVt2i+m3GPwbyBx9UsqsDxkXmd01ycFdpanodOxzZxOXSK5LNpMebFe3I
sjb4ICRZXWY1t8QynZ+bwXpWC5gCzxp7KsWVgQcf1v8/v9wbBOXk4FB7lfBYDz6r8iblpXDxqo4/M2y2SE80+i0+Rv8HsjVWZKk6sGUWyX5KH4UlJZ7knxaEEQ
yl0k69HrSbH+7vjbJNPRX7cXb8D6Y7sXbLZMJi1gEvZa0B8X+c7WD28Q3DDE1pz30UWnjkJ6DFAMpCmxN8sUNFWS7NfUh1g5ZCXwvKLWpnOE35w99VLvZhn
RpvOYEdEjW2+bDhSXVbR1zZrnMLoVBv98yNGYjGxJAHW1C7iyDVjE/+UTl5A71lGVUJFvYfZCdwcQMNGpZlKdJjAEOkpkcFrOqQ+A4Tqg9WpGvzAk0Q9ITMC
i1gLkJiztcVJmBkZsXSgGx5Y9UEvDXIGDn9QMQchDS1Osf52hY5P4WTCHUcERy8UlAhPa6Vb1nbtN5rP35PF07EFDE/uWZz6l3c1AoRx9xQ9e2FiwS0Ss+2
SrzlQL0wLUIaeBMSDNzOvWpkSj0CfMSXw3WgAp6bl1iREr+hD08wB0NFI4gcdKy6MiZVwwwMfy/KdIJxEaAWiEuTtfywtJHGaHYAkM8aWTBmoSmyYaK/ILX9
54emMfqZdRkyNSP7sE0eXoJoFXARJQWs2++bKH+Q8JJXPfsUgRAtdp3dxdIr1TJnHpv0yH9v1cRsa3eWjT5T6QYvUG9MrJ3F5GOHSVGZlyE7B5Dh8tVOpYH
+yXY+8tQz5Jc9Fn33F7zUGt5m+3lIDGVQOQqrzijDY8R/nz+uazBTNJCSkn//zsnBjJhAqGRA/3FFZeoaWDA+mZFWxLb5G26nDN2fOXGZZmHVLBjIWk6mB6
MhZ3Rm+7uMgdP0RdR9OZ1IzgQ/9XsAdrmDBmMTZbq6K2EeB+7U5ojMwcc6XLUyVb3l3AQUHSM8lf7WfcxK3Hhb7u+pL4BbUWjSBfUMASQMRVuzXhSgwp4HLsL
9UTeRQBwfFdJ4pD6+ofi7Da0nP9w/z1eC4ASJ4+Ji2wXOW8+ZqEkHiZH+GTHUb2Urwu4WXr0hw2L79qa3w+X2znaAdeX2XWGw6Cy2eLSLE8nY0tZx3O3OYcHw
uBGGUPpdKEmXsJuX8Xfk0kHOiwpUU3+jDtc3tZjEGSiXD3WiiJcX5s2fOPHRuUcmvHh8Kt6iK7SXU3YAmwPxg/QGaTeNw+Zsdh/UWiaw4X6aNUQWFwt56TTJ6f
pHb5vX0/8zx0qSbvVsQfpgvoGYgTUcqFrJgY5l0eSGjVyc+GBVoL4wSs7fcqyg4GKESa0e3h6o+4xHWAP+t0J0U2Bx4EEcgmbpl55iwkBFr0VMq4l9vbB8XHw
t8Uy0i2GcBxHNPEx+L+fjrSSxTLg3t1A+u+mc8pBcVJWeAzTnqN8NO2fwfS+KOVTY+VbVUV02E+L6YV+nZFqpW2hS6BIc+fI4y+cBTNo3JyHuSUG4E2PFSj9rJ6
+AbXF/0aoItP2R8C5zWMXJWvqPfZAPmLCUl+y+YORS0tXYcW4VOVHhSwwRpLe/N4y4eJSjB+jHpWaPDWCL1kzaVPqM9e2S9/M/2fDMHD9rZTdGkX9zKDG8Rk
XEjg3FHFvOVH0i5bP1xdNcJ3QyFyHgAsWmRVHIB9rhEgUG5oD3FRk5P3aLT/cmS5f0PWg8c3zFMj5hLgfdc0Swz3NZoLd4dzqhGFrm9iMPBY5fJXNGEEj
5XdKRx0H/fVAz3cOk9PfCS5PNuZvmsJn1cqOzHdtHDfUpqJtgDxvjE+i3nVYdL7LNz+TqDh1ebUAk4ZeGTR9ZUR/I1CjtL/hPBeNVbXaE7Zd1S3+0xg9QzOzrI
b0bBhVh60VXmbdKdRvPHs4wxVY/jhBUrTAJnrTLBZKEbLFAAPGkNc/gOfXjIzMoaHGYtP+cBDXtWuFzThJRpPNbmBwwdUvZ4+lyvkMCbhoTNgPx9xTUMk8l
4CNYRqbjdUaS7iT8h0u3dL3T5+KsyrfNwXvRk2M7lhBdmyL3oIgbBWLbXKxgeB69ULJd+xcg7nLBaPt6UeaOoJ7LTqBH7xwuTWfq4GTKeSwV2Y3eG5sa5Y9

V+bq6unrcLUHEYjG0Wi3jhlA6NRAIMOfpsx0SiQS5ubmMOE02BtFodERi8UBoNBrMmDGDZQHKysqg1+uhUChYCr+
9vR2tra2or69Hfn4+YrEYOjs70dLSgo6ODiaceCZApVKxQMlgMAwi2MdiMRw6dAiRSAQ1NTWoqalBfX39qCcZGo0
GeXl5OOecc3DxxRdj6tSpjHjZ0tKCtrY2ZkWRfv6SvovJZILZbGaBHtlEpD8eiy+Yx+NBS0sL035pbW3F+vXr0dP
TA4fDAafTCaVSifz8fKhUKiZDQcG7RqNhOlAUcBcVFcFsNjO+DVl7GI1mM3mCQ0MOY5jgXY6tyiVSjG+WFtbG2p
ra6FSqRgJWy6Xn5D8BxGxiRxME1CJRMIaQ6jkFIlEMhpKALDykFQqhUAgYCVR6rgltWy5XM7KVbQMlwUYaAlytuF
s3z9g/Pfx35J8XVpaytR1gUzy9d13341bb70VHo+HqSV7vV48+OCDePTTRR4dd56m2BKHBn1SgqTMmvd5PGQYyWUy
vd/f398Pj8cDr9Y6KYEl1CX9SVQe3Z5FtDgygtpFFEej+UTSFRMmrnlsvlLLNB2jkkL08DNNXfKSNGtfdYLDYoM5J
MJjM6cfx+PxtYpVIpFAoFsrKyBhnokdZOihbls5FoezMUF0kYrF4RCXu0cDn86G1tRVut5upVVVNHUbp4GvFN0n8
rkhmgDpH0TpFIJDJI0JCOIy0KhYJlBenz1BVD3xUnz1BVD3xUOh1m338l0GBFHJR0UrFA7/qnK0IpEImbqms6boQ5E4kqQivp
EYKjfA/gXqTs9QxmLxaBWq2EymVggLxQKB4kLEi+EfstUKsWulAGwDKGJ2rTQJYVxGMhgjsFnHRNkkVEuq/Z6QB
d/xqNhvFcyDMvGo0yDlEoFEIsFmPXODWcUGeaQqHIkDVIF12ksYq6B2kMo8CNhEMBsAnnUF23JC1BY65cLh8kj0C
g8TAcDjNrIFpo7CcH+6Gaachuic6/dCkKeo7G9G9+FAnmzpPKmB/w/3Gn0fdb7RuRsIBFg2m8zB08dX2le73T4qUc+
x4N+SfD1aSxDCH//4R/zmN78Z8aI+U8nXiUQCDoedna4dQ+UWn02VcmBMNnph8ZiASibDsF5Em04PGZDKJbdu
2ATiWWZo6dSpbCgoKkEwmEQgEBt0wCKFQCG1wbWhubsbOnTvx+eefo7u2UzioqKkJ+fnXmFMpMpweKkUgEfX196On
pybhhpGcG5uLiMXWywwWtt3hcJUej1evYZykCS+Wx7ezu7OSiVSshkMpWYzkNmSxDLt2MxyN7rNFooK
pmIDoQB9FamunhYi9AoGAlazpL4nmxqs/nmxQUiLMBSqWSKVprNJoM70SfJ3f3fKPPPYG4qabbsLf//53nnx
9PDgcDjz66KMP45z//GMz9R4DnmLn5vzsaUQu6/cuSPcsfvtRJYOVRLMVl2cA3B5oDpjdDSK3l/Fwf1sNJ1XdD2n
FRCKBJ598FvEvff/+Q33mmkq8lEsmkVXdLKTdicSZvI8SiQgtiQljSVlzXDkiLj8TgKCwtHJ1WoxOrdMfHHWqdIL1WoxY8YMzJgxA8u
XLx+XbSaQGzJtHkjjaJf4Xmd2KxWLmA2Y0GocVh43H4+jt7WUlV0I4HEHZ9fT3jY1E
mRiaTweVyDepKc7vd6OjoYJpBHR0daoQ1ipBmJlaBURvsDv96O3t5dp4VD7fHV1NTOnTodSqURlZSUWFiAc88
9F2VlZdBqtUz6gzK2ZFlhNptRVFEG0EnU6XsZ5AIMCyx5S5G0/EYrGMberrTiejjC3R3d+Pcc8+F0WhkzX4z
UyZieXfd4PAgGg4NKfrSQ9hAFc+nGxJT1A8dDrSJZXrUZVjWCYYUMDrVaLeC3jH4+jt7WUlV0I4HEHZDJZCy
LT6XdsQTelGke6Fk43PaklzaHW2giBYC5Eni9XqbZVlZWhsrKVhsrKStaVRhC0aVRhUOgUAAjUYDrVaLCy+8kI0vPPl6GLS0tOtC
iiy6CTqfD73//+oycORBzPxeBz
vf/87tm3bhscffxz0o3x+TFoLRr1y4899xz+2OxLKL5jG0nCgGy7x2eb
MmYPZs2djzpw5KC4uzuHDh1FUWeDwYApU6agoKBgkgyG41ACgqKoJer8cnn3yCNWvWYPfu3Wg2RvFu5d
jzpw5rFtSrVYjjJycH8+fPx5VNd99FJBLIQULYs2cP9uzZg2cVu/zZW96vVqtTzzzzOc
ZnClTpmDevHlcXHmYOXMmpNIpOZ/PSH8eCCC+++CLef/99jB49mnU+zN2PZ+FQP04+PPgP8fv9PMMHIBFhpsvjX3jx
n52PhwoU477zzWFKzZcuWlSAgqenBwcH4Pbt27F9+3Ydm8kQ8QdGg2ARsOoGHmnRsPnPbbthAzv0lVA/fC7HxdzTBQ
ccskluKCgYoGYNSHYDCYoX1HQrfpHm3Uqj9S2W0icDr5U2dEYPT9738fAoEAu7t2Lh48eIF7U5QGb59gKNzBSfL
5BG+//TZWrVoVjSs34tNp8Xm0uxb98+PPHEE7EjiRs34tPP8XmzZuXnn1cOCwgG/VXmdl6YgxKMvNeRt/ZwGsOGjbcOrHPgwBgpmPx
zWk6W78Jj8oLczE/1AHSmo6urC2vWrEFqSp0BGQHs/N0v6gpqjnf1EFgxMmnq4BwZcmtTxtzUsLcuSRvLlfA3XkmGXRC
XRdncnJwcVgqkm059ff2BSnrbZbLjoootw++ywf++23Y/13Z76mNnvX2iOTp1pwzCEnp1Fqzz1EixAs41zUPcMwBjC6EBT7SCKqPH
mzZuxdu1a7Nu3jp3U2EsjXb9++fj9++/CKysrLCpKrSEyyUF9y2l0oXqNmFaXdD31
lckGpVMJgMCAajcLpdDKeiNgOV1xMpFoTzJSuBYlYH29B/+dJCgmbyxNVu+ChY9RrDxm2SOTj7+KD3Co
EwsLCjHH8sssuy9+8QuEw2Hs2rULBkLmR8pvYtBE9+O7O8djuxFapqNcKAiBbAIMuj0ACHeCCLBBjbRfvNfvwXfAZDXgjBYfDxJATKK0u
vgkkfGLndbqxbtw6tPPvrokCn6gSncdJDD4mUQiweuvvT45OETEMDKZVVRUgXKJKHEKnXenJhqEYiKqiw0OABrE60Zb20c7N0htZvM3lTMQQ
lEjabDWq1ms0ssrKyoNNpNKdBhWAGWzFAnHi3H2A6M+Jcer2PUS2m7JL2nMK1UV0xuGFHfUCMn3vrk4mbTZrWfHdsy+sPfHamLAV8tV
GTVYloVCIERsBvosCALxeL3L3bv3H4QTrQIJkXUEZ9W0Fe1Uh4eDYPxKKfHgg2GtrhiL5KdTVCjazbWmvrPHvBLKQ4WTZ3UuHwnRr1AZ
9THt7Oz7++GOsRbMGVVVVa02VR4M1SaI2A6iz8cASXzoOzUgOdI8uDwVBMEvZWsKxDfCRAKvDT6L1Jb9fGKjfUTQY3W6E9Go5u6ZBm8qw
O5vF4HA0NDdi/fz/z4Egkk9jvrOoD7VZl4GJ+uWDsO7eTl1q5/FIrLmWv7qlLDO/BuDwKKN9w1zHeB2FLkaWNr0v5uZqLr6wsxMsZrjsK/v
z+w93DKnTq7S0FHfeeeMAf3fffff/4UFk1dtjt/vnmxMp6dOZXTi/2V/vXmB0AGP/dWrp+s9zxLwOLTJbjw6ZZz64dH8ZrSpZIe3P6nIi
44ALMnz9/1BpiJ4Pe3l6sr5N2LuhgH0oYqdIfsz4ejOiOkU6GJnrSqu57kTbzE66xhSdUCMPJuwJizX+HCjHFxC0xsXrftrfp0oo00c
KCgoAYFgZELFYjPnz52P27NmY7P49G2y5aB/yY2RdLOoGr6IYjZerBIFRXnsBhnK1qtqnVsG1ahfbFiv6jo1nt/tUGfJlbK5OpSoBAD93V6
E3njjDbzxxhv7X+fxuaDJOl9EkMtLbP+PbMkvx6Ny5kZrLLXtC1nBWdsa3tSHqDnI3KQZo8w+3KjPuAVz96NY8cSgaFU1dK36Pb9IIO9A
F7bffjr/+9a+45557cMMMM2Dt2rVqtv7ZA0lAhsTqqOq44JCjlqsgepHkUwbzv6L6dmCsgEoMUKQ4dCYmwQgO+NBFkYLQ7VEpSXE+JQU
0Rd/0mdvxYDQaMXZ5hzR4bMgj6ePxbk/KZ7VARgpqaHFCCN6ZtK1B1NhWhJQ0x1RGrIUBhlR1kKxyzppK91lrv+SLgs+rjf1Vm5Z1fDtXTkHV
KzjnnHMyZMwcajYsaNdrfa4o2ygDpWsTuI7C/6mhRgmrA7QXgbcoylfFnF9wkTeXKOwYRbaeEE4V+vMmWthoUzPy43Ek+DsS4UDGXVbUrX
cRCIRrFYrLBYN6OOKOLtHdPTw9aW1vR2tqKpoYGPpEUVVPljMM1lgEoa1yEqMGL9YbCXu3d9cfw5yzc1KFUXrG6Y1hcvcQTZSEnYBJ2DlS
qVNZpk+4xRmKNyWSSlYM0Gs2zsvTG99cqGWHbPiWdhBwSZs8hbcCpNZ6cNKXb2DmSyYMI0qKNzWSlYw0Gs2EDvw0MUifdKVSKbhcnmrFm6NLbT
ATU0N9uzZM8hcWC6X1KjYqyk9ZVpAXS7hRH6qJkDFaEYzzqBBBv2JHcaNv1Tz5gkrLgb1a/jMgVqbO4rQ4y2xq5EFdR3TcwEUWoVSKfRzG
YDNBoNIyrRL5e6f39bcpwr9MmJMbFvPFqT7prW2tuKjjjz7Cxvq61rLM8dDWCW35qk2ZT2K6kuTXLh33RqPThbOoKz3np2
44IIL8MILLpoMqYpM1yQBbAnGAgLlfn2KdevsWQSgu4rLLLsPn3/OTBYvvfrFS5ObmoqttDcjfMisOWTrSguH1+j
u7kZXVxxeCwSAb7H0HdHwALt37z5pE0yluUmr1YBBEzmMRqPo6Sacp6uCpNBp

l0vd0LBsbGwdF+QKBAGazmQ1gxcXFmDFjBnJzczNmSgAYCbOzsxNtbW2Qy+VYtmwZli1bNiw3YSSEw2G89dZb+J/
/+Z9hZ/yjRUlJCc477zycf/75OO+882CxWJhJqN/vZwFOa2sr2tra0NrayjR+Tub3F4vFKC0tRWlpKcrKyjBz5kz
MmjUL5eXlx70h9/X1obq6GlVVVi8eDG70QeDQVb6nT179pBdoASO4+B2u9HW1sYWr9eLRCLBzm8y+SULioEQiUT
Izs7GBRdcgKuvvhqXX345LBbLCf8m6TgbbOguPh1O1jxzH4eDBg/jss8+wc+dOfPbZZ3A4HCN+xmazYdq0aaioqIB
UKs0g0xOhnh4HAgHU19cPO9bl5ubipptuwiWXXIKShIUFhaeNcd0Io5hd3c3/va3v+HZZ59lkzeZTIYbb7wRd91
1F+bOnTusaPBEYLz3cSxdaM+MKKIXa1WIy8vDw8//HZaSJXqjIxhvSYyEYgxTjmZ/XqL/frYWdzAPqzEYY2jDcNxTbb
hbSyjH7tq1v8A1r27SNrSbCNRBYjOx3fywI7bBnYH2ERu3+wnrNDJU8bsp9EAxlNVhlKyLlIqIFGWDQDmlwhZNIzBHDv5UxmOmvjLNv3oG0lpDsEtTSmlOb4ZC0bsFhqVLbyNu/DsSXvO9d/qrQ3ipOqWXdQeFtuodm1z11nVtZKWlsw4Kw6lNU1obzIP7nh9dJSrfXxLX+DhbGeSjz0rnRrW0yT48tvRK8a2fNfTf6n7Pn/BYeRlMZc0l5a4G5/mD6+HF3EhFbF16q8fpekDKvqzwUvuaKvQCuF0DKwdCvqhZqmJCgx6Gb57SuzOwhsTWhqq43ihyXDL+7wDscLN34Bx60ohRWdb1GqxYT6MJDMyxdyeTGWcsMEQ7y8vKYQQdkD8J60cDDUL3RPZUI+x8XzNUqCVPbfIIxQsBwqrsw5I6o20o9/iMHGrthq71wXhVqy7JZbBW0LBEvJuCMQMO0xbfJhBxgSmo4JohPH2bDqGXzfCOyQVQnAaRpfWsHB61ncVIz9SNtdiRRLR5U/ozG1p+d8JjMsDv4SPvj8EmxN6SFpZgB2lnhjVMFY9BAAFTIbzCW4dmMZ4DdW8F03P9PwGwRYhO4MTv9k7ndQpWvkMRKHn5f7D9Z+QU1zjV36IUQWE42tLHaxTiTmDuzBqZrTmyrP5w6iG7L2Eqa+eeylpWqM2kPBmXp6TsCyXSwlSDbHdSl0MeLjl2YZuwNYOnLEMg+MaqYdFxEo40AoxFiwjc5DBqkKcOOs7lohupMqUfvm/8eoy6ZeR4eM7HGZ3EfAmqOZ7Mx9+3vq/mnwPUtxz2CMtyRQvbHCzkvWwldYg0ChWYT8ZqIYgOOYQ7eyxshwDIyIKjg6xb5/EW5YdB0NGXEiDOHs/3zafVQMpmNoyw8nyCWpzqc1yXuy+hsLWQwaG2L5hwQZ3lGrqnP4fg1XjhC+/v11LlkTcCoy/DJGBuQQpoMiZT9k3SFH7CDoMfiIHHDB4+FJX2mV7/d9FlM6LJbPYaGA8bQ6SKq9zqt6IDrwTQQ+7fBe3vGwBzIyw8dD8+cTvwWZ9UdtVCQgiFZpxdZxVpB0X1FW9XTnxtH8t1Epbv4jhIOoYDWzZG1GGEoGGwE3JXnJEZf0N+X4XaxkFLnTWqv2Wz7zBR4T04TRV5kS4vHnSmNe4tRDkgGxMt6mQQt6TIG7XBAIXT7c5uUc/gpXxOPrtmmRq3QvdS87FBDbnKlHsSWTxyrohHYGC/Q08rDcCg6X1gp4TEGowSmDmBr/2WMtAdDPtABvN0o6o7G66XWtTGSr6JlIs+yl4/QA1g4kJR/ulQ6XPl2lXdHkBXZ+AWt6W/4PqOiwqP4kCxKVu9MkI9WrDlqDB21SmYyrnbkEdg0PWRWAtSjS5n90jRtSTTWxbZl3MXwbWgLi/C6lvuU0zUXNaRaL3oEEB5Y/L5oI/b2NZMdqQTXRKrhFr/UyTgEPsTXcQqz4S4GIuZFmf+kl3R0yq04MaS8NP2tp0dmC7pXw1cQT/7+eqTZ5JmUbQhT9efyVTdzgOgQfMohhd4OUD/ItXnJwl1VT5EGSHyYpY6p/gAdWUQhN5+kpsnOQEEvI4hDYDXh3wqvaR5ZdU6XS2wuQNEI5+QS/KlKAHm0Dqj6ijqC9o0/9cKp0XX5TPTi2MNSL3RoDLRsnqQJLRQqpJyaldKjXTiN9A1J1l7J5fCW0uYc8EUvSQnkIAFYAwzDxdtY/XT0e8mNnTWYCwQyYaGDZz6ZzqsDcw51OpbhpgZ+tz0dKoG6jh9lPaS5YJCXqOAOA2AhyN56bKN6Z3LsfI7/HOj0+WEWP/Y9wuMQRt1PNG7+b0gKnfsDmUh6vDDH/DAjm0pVfk8JAYyl26hnfMTE/o5LG+TGmhJ4bv5Ky0rA0WB1ILdyCX6LIGaISEuxTrIgbI5kEgZ2DohRuPPl2r+ZsWbpUXhg1CvCMn

l0vd0LBsbGwdF+QKBAGazmQ1gxcXFmDFjBnJzczNmSgAYCbOzsxNtbW2Qy+VYtmwZli1bNiw3YSSEw2G89dZb+J/
/+Z9hZ/yjRUlJCc477zycf/75OO+882CxWJhJqN/vZwFOa2sr2tra0NrayjR+Tub3F4vFKC0tRWlpKcrKyjBz5kz
MmjUL5eXlx70h9/X1obq6GlVVVi8eDG70QeDQVb6nT179pBdoASO4+B2u9HW1sYWr9eLRCLBzm8y+SULioEQiUT
Izs7GBRdcgKuvvhqXX345LBbLCf8m6TgbbOguPh1O1jxzH4eDBg/jss8+wc+dOfPbZZ3A4HCN+xmazYdq0aaioqIB
UKs0g0xOhnh4HAgHU19cPO9bl5ubipptuwiWXXIKShIUFhaeNcd0Io5hd3c3/va3v+HZZ59lkzeZTIYbb7wRd91
1F+bOnTusaPBEYLz3cSxdaM+MKKIXa1WIy8vDw8//HAG+frZZ59FTU0NxGIxLrroIvz2t79FKpXCmjVr8Nhjj+H
NN9/EjTfeiFtvvRXbtm3D3r17MWfOHJSWlqK1tRW1tbWoq6vDlClTjrstPp8bDqqQ0VDzcWMGTNQUVG/sWDGeZMFwOIzt27ejqqo
KtbW16OrqYrNn0iehVl2VSoWioiUFBTAbDYzQuNwWYVUKoXOzk4kk8kMYTjgmFCez+djXRSkedLX14edO3fi//7
v/1BbW4tDhw6Nel/IiydKO9kIRKJsHz5ctx1111Ysy37ttswsLC0+4vEW//9GjR9HW1sY4Lv39/fD7/cjGvVi
77ruh1+vhdrtZacHj8UnCv10On0yE3NxD5eXnM7Z7avwsLC0+4vEW//9GjR9HW1sY4Lv39/fD7/cjGvVi
BwsJCRKNRrFu3DuvWrcOWLVvQ3t4+5HplMMhmmT5+ORYsW4aqrrsL06dPR0ZGBgYEBTNc+fHCzmIUbPzy/s2z//p/x7zsWynEKcfh/+9h+IXx5dx32c/xvdsPqu8uy/DPy7zXSKgOsLmLVuMn0Sa6bvw77SJ3/T+Pc7ZTXEaG
...

06dKT3j+O4+BwOJj0AcdxCAaDTD/P4/EwfzyqaESjUZZZIg854JgKfboUB/nQpcNkMsFmsyGRSLDJXXrmVSwWY9a
sWbxX2liQPusk0iFBLpcPaQly44034tVXX81Yz+7du3HkyBEcPXoUN9xwwyndh4E4cuQIVq1ahSNHjmDfvn0Z7e6
U0Rn4/0BjRiI9jkeJj+M4ljan2ZPf70c4HGYzcKlUmjEjEovF0Ov1UKlUrOxA3I1YLAahUIjGxkbmBZSejRtYkhh
uoczQwPJAMplES0sLjhw5gubmZsYpIlNFWoB/SflnZ2fDZDJlGEKSkzfV5UmMsaGh4YRbiQkSiQRTp07FtGnTMGX
KFGRnZ0Oj0TANKhpIgsEgy6KZzWaYTCbGo0gn5ttsNqZ9pFQqmYFqTU0N9u/fj8bGRnR0dCAcDqOsrAxlZWWsfOZ
wONDY2IhQKISsrCyYTCZmY1BcXAybzTamQSQejzOLBa1WC4PBkOGbdTaDSgkkXzEZAo1wOIyDBw/C4XCwmxbd1ML
hMNxuN6qqliZZGAJLSsrC9nZ2VCpVBAIBBCLxayhIB2pVAOoOhwOtra1wOp0sw6NQKBhnJ/0vZaPTy3ZSqZSR/wc
2EtBCfB0eEwOBQMA6QccbPp8P7e3t8Hq9zNZmtFwm3ittBLjdbnz22Wf42c9+Br/fjweeCCDfL1y5cphP0uWIEO
BUp4jzd5PlSXI3r178Ytf/GJc1qXT6VhpIzs7G1lZWUxJOx6PM34KEZ5jsRgLfILBIAuETrTCSjyDiQIFv0qlkjm
Sk67SRMNsNqO8vBz5+fmMZ0AkSCoj9vf3oDBKZDG63G06nE7W1tQgEAqiurkZ1dfWEb+dANDU1Yd26daN+v1A
ohNVqZUrEFOR6vV74fD6mfZVMJpnjSwwnswclDnC5lcUjqfyrcU5BcUFKCyshI2mw0CgQB+vx+HDh1CXV0dUxSni41ACS
TSdhsNtjtdggEAkQiEUqjVauTn57PJUXqphcov5dtmsVggEgqQFoZBw10p6im7/JZEJBQQEjllmMzRCQSgUwmg0KhQC
YUVcZdVe5XC643W64XK5B3Cy1Ws06EXNycph6eim7/JZEJBQQEjllMzRCQSgUwmg0KhQC
QYEr1PT098Pl8bOJARPPm5uZRXQ+PPPLIqM8JAKwrTa/XM0M0XykWJUgkEhY8BYNB+Hw+Hw+No6IRKIMFXMqldP/xHP
Ky8tDfn4+8vLyYDQaEqWYckZTHcbjfjsAxs5KCSlVAoZDy/QCDAjhH9tn19fUyQlOM4GI1GWCwWiEQixRCKRDIs
kjUbDxkSJRMKuBeLyEB+Nxo+RlvTzmcby559/HgaDAbm5uUyLzmAwQCAQwO12s4xPzVVmQHrdzacQIzuFGa/ZaUs
DwcCEfJ1OJ6LRKGumoa5YKnNS4Ev0DZqAhkIhNqmn7zAYDMjfz86FrSqFQIDc3l92D08cQhULBlMZPhyX
IpA+MGhoawHEc5s2bh69+9at46mqmnhnxfbm4uLr30UgBgXWnV1dW45557hnxfbm4uLr30UgBgXWnV1dW45557hnxfbm4uLr30UgBgXWnV1dW45557hnxfbm4uLr30UgBgXWnV1dW45557hnxfbm4uLr30UgBgXWnV1dW45557hnx

GREhoI1dYstQnFdtSc7IWAoVCAZlMxhfPhYJy16gKJHRjVSgUkEgkBaMCstks5yElk8mcwZvwAC8pKYHRaERZWVl
OPpzVaj3prJ0IoTT7J6LrUttXC0EgEIDL5YLH40EoFMqRrwvdyCnrqNAiVPzQ90HkelKllZeXQ6VSIZ1OY2xsjKX
VFHZqsVggk8nYMbi/vx9DQ0NwOBw5A0UhyAk937CNTPnIXTwQCLATNxFsY7EYtzNPhnyHenJrV6lUkEqlfHOlAFz
iDsRiMW6f0ucIvbYWCyLc6vV6AGC7AFJoCsmpNMHwer05k6mzWH6QAlIoFMl/VCgUcLlcmJychMfjgc/nQzAYRDw
eRzqdnlWJJeuG/KgS8jnLP+eWckwVFRXl2FcUFRXx+9KAnK6T2WyWBQxAblWerk/UWSgtLeWsNBo4ZjIZ5kGStQB
dWwCwHF844CTLgpmZGaRSqRyVrjDnjOJCSH5Pi9/v52s3gar9xcXFbF1BNhhisRharZZVknQOBQIBbN26FTfeeOM
yHC1v4x0VCUL4y1/+gquuugpmsxnT09OzIkFqa2sxNjbGf08Hj0KhwM0334zbbrsNX/rSl3DgwAEMDAxg586dePP
NN/Hqq6/OSVY+k1RpwIkL9/j4OMbGxjA6OgqHw8E3CuDEPqEqCBk0kkeI8EJTKFBRSC4m2Xd+hlf+olKpmMy3nIo
tGpjRYGCt4J2gShO2kfx+P9LpNFsJGAwGZDIZ/PnPf0ZDQwNkMhmsVutJo1II6XQaHo8HDocjJz8ukUigpKQEer0
enZ2daGtrg1QqRSwW43Di5dq2SCQCu90Oj8cDhULBqhq1Ws3bvnv3bvz1X/8130QW+xlutxv9/f2YmJiATqeDwWB
gywqXy4XR0VH4fD6+gVNUw/DwMAYHBxGJRHiQTjdm8qQhUPWJ/j8UCkEikUCtVqOmpgZbtmxBbW1tjhKQAqKlUil
GRkZQUlLCSiS6WctkMo4VIj8zEhv4/X5Eo1EAb1fFmpqa0NTUxO1XiUTCCjZaSF1YVFTEEwGKICKZPi3BYJAHLXT
dIBfmtTbozA8CpuqWUqlk2wOaxJ4J0Gq1C568rDSuvvpq/OIXvzirSpsPo6Oj+PSnPw0AuOSSS/D1r3+dI0Fuvvl
m9Pb2AgDuvvtuPPTQQ/jgBz8IAHjwwQfxs5/9DB/72McwOjoKk8mEf/mXf8Hdd9+Nl156CU888cS8cS8Cq4zTZUmlUr
R2dmJzs7OZV6r2Z9zKv97qvvudAw6fD4fjh8/juPHj8PhcHAVo6qqCrW1tdBqtRCJRHC73RgdHWXO13JWFlcLJME
vBOJ8rV+/ftHfg1Qq5YreQrAcad/5OJlyUqFQcPtnqccZGawuN8hhWyjZJqRSKeZVLeR9lqq8i8VimJmZYeuG1UY
kEuFBFBltFvp5enoawIn7RVNTE6xWKyfL0+BNGPibTCZzvJyEi5DPRwsNSmngSgPOQqDBHcnj6TEej+e0+YXVR2r
pk2UKVWLoMR6PY9euXWhsbITdbkd/fz/6+/vhcrm4FW2xWHLsJEhMIhKJoFQq2bqC2qAU9ko5joUGofl+RJRRSdS
N/HZhMBiEzWZDMBiEyWRCWVkZmpub0dbWBrFYDKfTiVAoxANnmoRVVlaejQQ5GT7/+c9DIpHg0ksvxUsvvQ+Qq1Qq
VSoWOjg4ut5HiSCKRoLm5Gbfffjuef/55PPnkk/jYxz6G2tpa/OAHP8Dg4CDgGxsawZcsWXHvttad5y959SCQSBWc
l+Rd04c8qleqkN8lIJIK+vj709PSgp6cHAwMDfKGkmwYtcrmcI0GEJoSRSATZbJYDUkOhEKanp9Hb27skjg9wQlX
Z3t6O9evXY8OGDdiwYQM6OjoKqjTyAyfP4iyEmO8GsVqVU6osnS5Qy/lkg+uTDf5InHOqoPbyfBBytJYLyWQSg4O
DuOKKK5YlK83v98PpdEKv17N4gdpbxB1TKpWQyWTMq6RW9krhbCTIPPD5fHjmmWdwzz334gMf+MCAAAD
ffTfWr1+PVCqF//mf/gBAMQnkQu50OjE6OopoNAqdTgez2Yxzzz0X5513HqHqqqqs4OzM5iSSAC8kIy4yiiSOjqTz+
TAlEikcxysKc23Jnaql6LEIlEBcOthakAQkil0oKc3HcS1vzAiCJBWltbUVdXh4MHD+Kee+6Zlk7GROp9N49NF
HZ0WCOJ1Obrs9/PDDePjhh/l3hW6qq6VKs9vteOWVV5bt/fJBfCIi4Z0KksnkipK85wJJ2YeGhub8m9LSUrS2tqK
1tRXNzc2wWCzQ6/XM76AlFosx/0EmkzHpmWZ+wWCQXYbVajUaGhrQ1tZ20r60kGNEOWpjY2M4fvw4jyH49imPHjuH
o0aNwOp3zKtMo1uV0gHLQ6uvr0dDQgNLSUj4PaCGPqGw2i9HRUea0OZ1OqNVq6PV6xGIx+P1+HixKpVJWotHS1ta
Gc889ly+yMzMz6OrqwsDAAIeLkq8TEV+z2SwTTckVGQATQsvKyritSSRaYd4U8eyEBPtMJgOv18tE1YmJCTz33HN
wuVyw2+1wOBxMFCXyt3CRSqU5KeM6nQ4NDQ2wWq0cEkwTEa1Wi9bWVpSXl7NzM0XQWK3WRbVeY7EYIpEI2wgkk0k
4HA7s378fe/fuxMn3/3udxCJRjenkYoFGLbD6oa0AC/qKiI24OlpaUoKSnBxMQEjh4
9ysvQ0FAOaZ++P8ohyz+G6LhYKvR6PSwWC8rLy/mxrKyMn2u1WnR1dTEh3ul0Ynp6GtFoFJlMBmVlZaiqquLWLp3
bc1mPnMyaJP9vSHxAbazlxlpTpYXDYTidzpztptgnUlISv5DC20mhKpVKZ/nHbd++HX/3d38H4KwqrSBWIhLk4os
vxsaNG3O8jwphNVVpAwMDObljQvURhRLSVyVU3ABgYjRJ8WnGRcn2+SCyJR3EFMQorAAJI0LoZ4VCwUGJQoWU8MZ
FRFryXNLpdCguLuZto3iHQheLuQ7FaDTK20VEU9oHpByyWCyr4rZMqo1IJMLHGSlN6OZ4sopLKBSCzWbjfr9EIuH
9RdESRJrNvwkLX8t/HovFoFaruWJosVigVCp5EEgcFVKlKRQKRCIRuFwu9PX1YWRk5LQQL+nCuVxBwGKxeF41Gvk
VEU9HeDydTohEIo75oApKIpFAOp1mTlsymWR1plCxVFxcjGg0elr92E4FwggXGsgRl0a4rIXvabEgywVhCG9xcTH
kcnlOjl3+Y6EQ5PxBvkqlglarhdVqRU1NDQ9cKV+OLFcoJ48oCUQvEFb8i4uL4ff74XK5BonVETTtUapVHIUDV1
b8sOUTxUXXnghvvjFLy7re76jVGk+nw9GoxF33nkngnVRVQA40UKrr6/HrbfeylEge/fuxR133IGf/vSnuPfee
4/X7odDrcdddd+MIXvzirSnOQ6DZ/PB4fDAYfDAafTiXg8jsbGRjQ1NUGj0UChUMBjsSjH/0oVCoVbrnlFqxfvx6f/vSnOD
Z6OSPRqPseE1KBaPRuCC1UDqdhtfrhcvlQigUQiKRQE9PDzQaDRP78pUWwucULjk9PY10Os0lWWGJViqVcqmcAi5
tNhv6+/sxMDAw7wAGmC23XShIFq1UKhGLxVhhApwYkJEpI+HIkSP4xS9+wT/+4z/yzne+k3e84x3ccccdfOpTn+
ufE7S5nQ6DZ/PB4fDAYfDiTiXg8jsbGRjQ1NUGj0UChUMBjsSjH/0oVCoVbrnlFqxfvx6f/vSnOD
AAADgiiuuwCOPPDIrCuSjH/0oVCoVbrnlFqxfvx6f/vSnodPp8NWvfhXGxsbGjbWvfhUqlQp///rOfxcD
Z6OSPRqPseE1KBaPRuCC1UDqdhtfrhcvlQigUQiKRQE9PDzQaDRP78pUWwucULjk9PY10Os0lWWGJViqVcqmcAi5
tNhv6+/sxMDAw7wAGmC23XShIFq1UKhGLxVhhApwYkJEpI+HIkSP4xS9+wT/+4z/yzne+k3e84x3ccccdfOpTn+
ufE7S5nQ6DZ/PB4fDAYfDiTiXg8jsbGRjQ1NUGj0UChUMBjsSjH/0oVCoVbrnlFqxfvx6f/vSnODZsJWjGQyGSS
ufE7S5nQ6DZ/PB4fDAYfDiTiXg8jsbGRjQ1NUGj0UChUMBjsSjH/0oVCoVbrnlFqxfvx6f/vSnOD

jenoaOp0uJ8x3Lqz0NpJB5EpAJpNxy3Q+rNb3KBwoCX8mJ/1EIsEDZqfTySHHFO5N1zPh5EMul+dc7+i58BEAXnr
pJXR2dsLj8aCrqwvHjx9nVZpEIkF5eTmMRiN3B+LxOMLhMHOHkskk80n9fj9mZmZgNptRV1fHLWnh9YYSF2hgKpP
JUFVVxYuwYk/bT/sAwKxMyZPhbFbaIiA0uSoqKsq5CFAkAPB2FMgf/vAHfPSjH8WmTZvYFyOVSqG0tBTbtm3Dn/7
0pzkHRquFX//617jlllswPT2d0/oSi8U5gwBa8l/L/5lmhUJSLgAebFD8xFyPJytXU5KyML6DngNg6ezpnlkVFRW
hrq4OarWaydDkUj1X64ZM2IQz28bGRmzatImXxsbGWSd4KpXCzMwM/vjHP2L79u2cMu31enHo0CHs378fvb29GB4
eRiKR4CTwQggGg3P+biVRW1uL6urqHFfqkpKSWeZ25PhbVFSE0dFRdHd348CBA+jv74dSqYRGo2GzRRoAqFQqWK3
WnAtpZ2cntmzZgrq6OgAnbixEHqeWivCcJbf2iooKWCwW5hgJzSQVCsW8UnlyKl6pG/dCQDem/NdILLFYLPX/zuL
UIOQVFUJ+Zb6+vh7nnXfesnx2MpnE2NgYduzYsSaJ6HRPOJ0KxlPBmh8Y+Xw+vPzyy7jjjjsQDAZnka5/+tOfzvm
/+aq0TZs28fMDBw4AQA5pOx+rRb4m47d8ZDIZvhmdLhB/gcrYNAAgz4uT/S85ItNNjG52pLITqu3yXyPllVQqRSK
RQCqVYvdm4mCUlJTwDJq4RlQhqa2tLXjRyGazCIfDmJmZmTUQWkiVjhxx81FcXAyDwYCampqcz73iiiv4eSKR4Gq
Oz+fjSgblf6XTaQQCAeaICTljhV4nAi/tQ71ez60Di8WCoqIiDAwMYHh4mHlIBoMBdXV1KC4uhsfjwfj4OEZGRjA
6OorR0dGTbj8APPLIIwv6OyGGh4dnvUYzW+BtjtxCIBaLc74DcvFtbGxEfX09SktLodPp+Hug/SOXy2GxWGAymZB
MJhGJRJicS22ygYEB7Nq1Cw6HAzabDXa7HWKxmLPmhPYORL5WKBS8761WK9ra2lBbW8sVTbKOUCqVaG5uhtVqzXH
OTqfT7FpOYcQ0GCQrEo1Gs+AqMokkqPIk/L+1QNwlEYSwArKcWAvbuJJ4p28fcDYSZF4sdxRIZWUl3G43UqkUvvG
Nb+Cuu+6a8z1Xi3wdCoXg8XhYGUVZTUS2E5LuFvIacWOEBGma2eRHUOQvxcXFbBRHJG36XxpQ+P1+vmAL22H0nAz
QyFfqLNY+QqEQxsbGuC1DCxHMhQobh8OBiYkJiEQiWCwWlJWVoaGhAdXV1VyZI7UfGcJFo9GclqfL5cLQ0BC8Xm/
OepB0GACr0Mh0j1ph09PTZyQJ91RBkxTK3SLnemp5EFnZ6XTCZrPlGPEJibeZTIZVY6SyKyQkED6nth59VjQahc/
n4++T8hepCitUamUymZwJDU1yCBTsSwRg4vjRpEXYaspf6HdEbDabzWznQU7nfr+fXbtFIhG3q/V6PR/TiUSCLTc
okoZUf3OJH6haSW0qYWwRHbvC26tSqURpaSmbY0qlUh780vFdaF/lL7SfKdhYGF5MfDGK96D9S/cWqRS6bLwHCk
sOhwO80CXBuNyuXxWWLPQiJNy8lQqVU4MSzqdhlKpXPaq7juKfL3cqrSRkRGEQiG88cYbuPPOO/HDH/4Q119/fcH
3PNPJ12sJ7/TtA85u41IwOTkJp9PJ1Zz6+vqTigNSqRTcbjcT6tPpNMLhMFwuFwYGBjA2Nsa8iXQ6nXODikQicDq
d8Hg8kMvlXOWzWCyQy+UIBoOYnJzEpk2bUFVVBavVioqKCmSzWW51C+XvFGtBAwWn04mRkRH09vZifHycI0GMRiM
MBgMCgQD6+/vhcDhyBnwikQh2ux1jY2M5M1uiC5wpsRJrAWKxGCaTCX6//+x+mwNFRUWzMuaIr+ZyuXgyPd9CauH
51J/kGi4SieB0Ohc8obn++uvx8MMPnyVfz4WmpiaIRCLs27cPu3fvnlOVNjk5yWGqCoUCtbW1aGho4EHRiy++iJ0
7d856/6985StzDozOtEiQMwHv9O0Dzm7jYlBXV8cco8V8NpljEkwmE2pra7F169ZTWp/VIO3OB1KGEYeLZtxkfEg
VPRJECFuzFNhstVqxbt06WK1WrnrQApyY5T/33HPo6OhAKpWaJSSYawkGg3C73XA6nnVCpVDmhzWT/QPtMKKQgd2i
qJAifJxKJHFPZYDAIpVLJknPhupO6Ln8Jh8Nwu90YGxvD+Pg4EokEnE4n71ONRsPeRlRVo7w+IWg/ZLNZiESinLz
I/EfiblKFR6VSsdBFr9dz21NYMaEAVcqPo0BWku0L1bPCJb/iQ/ekF154Aa2trXA6nejr60Nvby9HWg9X76hqRlU
s8mcDTkwultMvjap+xEGkYyBfiCL0xIpGo5iamkIgEOCgWqIWCNvgZ8nXBbBcqjRCX18fjxbvv/9+PPHEE6dlu85
idRCPxxGNRpFKpXLSpReDbDaLqakpNrMbGxtDNBpFMpmE0WjkdpLRaMTx48dRXFyck3xttVpXxOTtLN55ILfnfMj
l8iWRrKlFJUQymcTQ0BB27ty5Jgbwi1XNzodMJgOXy4Xx8XxXEcPHgQ119//Zz+ZkIqglwuX1OB1CdDMpnEwMAALrr
ookV9hyTAEbb9aAmHw2ycSbzS+Zbi4mL2qhOSrDOZDKLRaM7npNNpWK1WmM3mRfHkThfOnCPh/8dSVWm///3vAYC
Jqnv37sVPf/pT/MM//MNp2Q4hXn31VXzve9+D2+3GY489hvLyctTX1yMajaKsrAxHjx7FoUOHUFNTg+bmZjz/
k8ZJSy+12w+12I5FIzOr5UgnUZDLBYrEwP0EkEnHfe74lFothamoKv/71r3ngnqUnrR9tKS/zMpw0wmE8rLyzkElFyfqWQvFotZlSY
k8ZJSy+12w+12I5FIzOr5UgnUZDLBYrEwP0EkEnHfe74lFothamoKv/71r3ngUGgR9r7zH0nlIJw9CdVf5G9jt9v
5OQUqEjQaDe+fsrIyvhjmy27T6TRLpHt6ema9z2KgVCrR3t7OHADiN5SUlPBnEUeLngv9e/Jn2XO9JpfL4fV6MTk
5yb4zoVCIt5cCHh0OB0ZHRxEMBlkm39TUhJaWFlitVuh0ukXxD+hGOzExwYZz6XQ6h5sh5COZzeaz3LN3KQo5mdM
1im7wQnFHIU6RTCabMzBXLBajrKwMpaWlsNvt83JB6XPfTSBO2kqa4ZLxpEqlOq3qz1PBmj8qlkuVRje9c889F8C
J6INPf/rTuPvuu+f8/9VSpQ0PD+PJJ59yctvc7i7lBJoD5xoIng0QiQUtLCzo7O9HY2IiIUFRXB4/GwaNHjx
X64VarUYmk8HUxB4HMzAzHQezfvx/79+9foa1aXshkMq5OqNVq9iKh414sFuOXv/wlwuEw+vr6MDw8jMnEOmPRmIA9LW1oZ
zzjkHdXV1TJQdGxvD1NQU3zCppSRc6DVh6b2oqAhKpRJmsxkmk4krALSQAgwA84MikQgCgQAmJiaYeyiTyTA4OIg
//OEPOQaCpHpTKpWz9gepJYUk4Lq6OpSXlyMQCMDj8TBRuaSkBO3t7aiurs7xekmlUjCZTKipqYFGo4FCocDQ
oNNDr9chkMpzQHolE2BxPq9Uy0XhwcBDd3d3M6cif6dOM/r777kNRUVFRUOW0cowhAKNYQ/FxcXIxAIcGVmeHgYU1N
TiEajHHtiNBohkUhynPGJ5L2cxyktOp0OdXV1qK6u5klHX18fXnrppEajHHtiNBohkUhynPGJ5L2cxyktOp0OdXV1qK6u5klHX18fXnrppVlu/el0mtteRqOR/aCEQdP5Cx1DwskekYQ
py1KoMMxmswX9iQoRyfMrKNlslid8Qsdp4YAyHA7j0KFDbJpIRqQrPeEgdTL5M9EjpR6QQzuR7onDR4T2YDCI/v5
+9Pb2QiKRsNM7rTe1R00mE/upnVWlFcByqdL6+vrw8xsv45xzzkE8Hsf//Hsf+/Hsf//u/+45xzkE8Hsf//u/+45
z4ejRo1w1EC50IQMwywhR+BrJ6IX9euon5yM/kJEW8j4ipUZ+ZYoIqERCpew14U1qrkfiS9DNQGhtn78AyKneEIh
DJlxIuUKyfeGi1+vZfVksFiMej+cE5Pr9/oKJ9lSVpBtBeXk5KisrueKyGKTTaVZxUZwJ3diI4FioEig0iBOqUYS
Bm/m/A07cLMi6gMw4aVvpWNFoNBwunEqlOKLEZrMt2X1ZoVDAbDazIjI/M4mOnZmZmXelouwsZoOO9fxBtUQi4Za
30LbgnQqKDCHV5VKjT8j3DlhY5ltRUdEs7hI5pcdisYIxJYuZABHocyKRyIINdS+66KJ5UymWgjNKleZwOHDPPff
MSaomVdp5552HoaGRvdp5552HoaGRvdp5552HoaGRvdp5552HoaGRvdp5552HoaGRvdp5552HoaGRvdp5552HoaG
K5uZm/OlPfyq4bme6Ko0GUCTjpxH86WhjnFVsrSwymQzi8XiOvcJSEIvF4HIvF4kMzEM5kM9uzZA4PBALvdzuP
BALVazYG9FOp6MqTTabjdbuZs7d+/H06nE2KxGGVlZTh69CgOHTqEmpoaNDc34/nnn8fOnTtRXFyck3xttVpXxOTtLN55ILfnfMj
M1WpFXV0dy9DHx8exWCwGAgEUFBTkbDabzWazWCyGXC7H0aNHcejQIdTU1KC5uRnPP/88du7cieLiYpjNZlit
VqxduxZHjx7FoUOHUFNTg+bmZjz/k8ZJSy+12w+12I5FIzOr5UgnUZDLBYrEwP0EkEnHfe74lFothamoKv/71r3ngnqUnrR9tKS/zMpw0wmE8rLyzkElFyfqWQvFotZlSY
kkErhcLoyOjiIcDuOee+7BD37wA1x//fX4/ve/j+uvvx4/+MEPcP3112PHjh0IBoOIRqMoKyvD0aNHcejQIdTU1KC5uRnPP/88du7cieLiYpjNZlitVqxduxZHjx7FoUOHUFNTg+bmZjz/k8ZJSy+12w+12I5FIzOr5UgnUZDLBYrEwP0EkEnHfe74lFothamoKv/71r3ngnqUnrR9tKS/zMpw0wmE8rLyzkElFyfqWQvFotZlSY

+Vl+hx5KSEpjNZlitVjQ2NqKqqopl4tPT0/B4PMxDIal/futX2Maiakwmkyl4DFOlIn9CQBULr9eLkZERTE5OMj3
A4XCgsbGRt0/o6eT3+9lfir5XYdC0sPpDP5P7Py1isZi5NAD4GKPrq3D9hNWVxSjlpFLpLCK2UqmEXC7H9PQ0xGI
xZmZmYLfbly1rcDEgNSlZAFA1i6wkCg1oy8rK0NLSArFYDKfTyRwk4G07mcsvvxzf+MY33p2qtNHRUVxwwQXQ6XS
zSNU333wzOwUDgN/vx2OPPYaGhga4XC78+te/xl133YVf/epX/H433HADbrnlFoTDYXz/+9/HH//4R4yNjaGurg7
33nsvfvKTn+C///u/0dHRgf379+PjH//4vKz8s6q05cc7ffuA07eNC4lqORko7LKhoaHg7+nmtVTVllQqZefr7du
3n3bX+Xwkk0n8/ve/h8FgwMGDB3Ho0CGMj49DJpOhrKwMKpUqhztFfjvUfmxsbERjYyN27NgBi8XCpplarTZn0Dc
fKBqEvKKIQwicmEgODQ3x4na7o7odfrYTAYEI1G4Xa7swWvXrkz+6KiIpSVlTG3TqFQ4IorrsD27dvvXjDFMfvYiq
2p1tdOB+oVZY/YIrH49wSpsHvXOuev32pVIojgPLjleZ6Ho/HcwjS4XAYJSUl3IoUtv/men6yyYzkej8PlciEYDEK
j0UCn080SA8yFd20kyOc//3mW4GvVQh0dHbjxxhsRiURw6623wmm2IxgM4rzzzzoNKpUJdXR22bds2S25YUVHBlut
tbW2orKzErl278JnPfAavv/46alx11VUATrebvttMyys7H5OQkXn/9dbz00kvo7u5u5mPoPQ0Zl64WvvtJD+
Ls1jLiMVi3DoVmsvNt0xNVmsvNt0xNTa3YdUEsFsNqtcJgMHA7lbyPEH4qFFGwYlkuVaJRqVmVVxbbmwS3CeH3HX9
PPPEENBoNampqUFxdJ7em4ymaBWq2EwGGQnVFRqMv8PrQmnZHeVmwqhJ4ObnMMrEzsmh4EiRGmsGvBtmMSNeyVJ+L
k35WxsTF+HgwG2ceGFFzknULSa6vVis7OTtTU1Mz6jqPRKHHp6etDd3Y2uri54MMoZ+xMQiaTQU9
PD/r6+nibx8bGsbGMDAwwKKHBFKSqVCqZb3esBCtra3s6G232xGNRme1Noj7ZrPZ4DiI
Wi0Gr1aKhoQGBQIA9dvJz8fFH1CakWTyYSGhgZeysrKMD09rOa+1CAMJ70h5CAQz2Xm0vfvU
rHDt2DIFAAMeOHcOxY8fm3R+lpV8XRE+1tTUQKvVnr3WzIFsNvGn7SQ7xQtdI6uW7CGra2tqKurg8lk4qofDUw
LnbOZTAY+n485Q8LlTvH1xJO28BocHAQ2WWra2tc/4MN+RO9973+vxcGDB3HPPfMfMfMUqU9+OCDs/4vD4r+MiQ
SCS666CIAwHvn8x68+uqruPzPy6w48Q2RmyNzq3JCK7vRcZDGJZn+5iKRjaMTgqZ7z5ofEH7p6tqKurg8lk4qofDUw
6VFdXc8WNyK+UWC+Xy7kPnEqlIqGRM6vUarUFrfkLLLcQXoX13JiMQCGD37t149tln8eabyK
RSHB5mkzTQqFQIk7delkivzIZVKUVpaCoCOCo1Gw7YU3rYPPpwYU7
wHhsbAwBW5BEfad26+ftVa29tln8eabbyK
T8YXE/HyivsFggEajQV9fH66Ce2XP9f6Cwm+v/dlpRo80Owke0qamSHthkJ
IJpN46aWX4HrDqamBhdddBFGRkYY2oZ2Ezx0uP9SR+/HF6uL6vvh3rD
jeTYptSorK9Hc586d2d2LZt20ndzJcbp5KzRa0uquhks1k4HA7s3r3oZ8tM0
p0YCHuHoLISy/+eab+PWz3n74uKinjiK5VKEYlH8tbFcC8Abcu
NpN6VWWkLifq49957ceedd8Ln80xpxk+ppyxSCSC8vJy8JjvvvfJjEJz4BAPjl
8GLfffjvuv/9+fOpTnrTyzr4vqulSlsIKPYgEAhwcGj+cyJhTk5OrikrfKlUCq1Wi9Li9LSUjQ1NaG1tZXdcomcomwJxKoRT8
qZ52clL1FAxAiUi5ohirMdxOLxTCbzQ0bD79+/n0t6NDQ1Bh6z5vBDmMjs6I8Ym/t9PIbyqPKkKaRgmsz
T7emCODk5OecFV61Wo7q6GN0rm4qzA5Sk9yqRPh+S7SPKzJDFKexEwbgJJAOKu3bqhmM
qLy+HRqPhfUXkZLVajeLi4jV30T8T8VzMzM4MiRIzh27BhsNhtbKwDQTgcDvT29sWCWUlZZXhwgsvxLZt21BdXYhQ
vWycajcJut+f4f4g5Ff2EIgCQIbWWULIufE4hvzQQTaVSX+Pg4pqen+Zz
Jz4Kj6ys5aRMJXK1WIxAInLLNABHjCy0qlYqJ99T6DADaVA9WI0RcpVKhuroaFosRqMR9fX12
qzefzwWg04s4770QwGCyoSgsEArj22mvvCOP4KmnnsLLL79cLyMf2d+9evOc97+H31Jk
f/exn0dPTg6effrrhaJSrHnMlip+pqrRkMonjx49rxJuMTPC
skgIX88mDtCxl8EBtIr1eD5VKhKhYmJiYIeORKJBDU1NaiHahogl8tzStVUxaDUdIJIEJNTQ0uuOACXT
RRdDr9chms3A6nRge0FbaFfG0KobGxEVvPqSRVdee5SV27tzJHTWFvqHtJ+JaAFgK1dijAkwCG
cayCQSCQwPj6ndFR2O12Vu0VuvQ+Q901fX1/B/9B/9NqtdeIPBa2towxFBOEcJKzcjZbb76JwcFBO
F2y2UEFMUQktLC1paWqj2j3h2lZWV6OOO+7ArfPkl1VXOEOHWa/cGe3Lxr+OO4
FgnPPPPRfnnnsuSuNmzNmzYdIGow8PDBCRO+5Ki7cYmJRzzqPaxaDUcQIQgiSKcqXvhv+CPkUCUXtjBVr+jFiE
2u33evxeJRKitrYXFYk6vR0NCAyspKpK/nT9uiLTSdhzkR13UXMgIDiN84QTLp/ycKbedvy
rIRKJOFJiqS3R5YJIEJ7ezuuvvpqXHLJXzeTOxMYHR0FIFAALFYbNnMn5QYO9U+ETxeNxtt/1eDxIpVKQy+Xo6Xo6ur
C1VdfjfLycgCYpdJMNsUNvYXu7o9PRw9AgpOcnKIx9SqRTvec978P73vQ+HDh3
c/+KLL8Yrr7yC5C5uZmfPOb38xwoBYvsPxCiAbXr+cc2v//jBLpVTKXHTTtuPnEDTjJxWDDtnaaJKs3gGpxxnsbGtW
XvvKVr2DPnj4w4z0SI8kFqRdYXpcuyqaxh+RfR3YM3g+9Lk9bqPFDtQxMXq3hhEeHebO7bgOZj
mpvDb3/4WsVgb731k37ZKllVXJjnmuwUgikQiVlZU8+Nq4cSUoOuvPJKNDY2ALT1unTswRLxN
sbAxOpxMKhQ6QKXXnopzj1pG7PZLDwej5uWd2TJqEwZ4ozZgvwqdLB5Xj59pQ1Bxh0Y
oCFVVVRgfH8+ZJFCWwKrQ21tbdB4YYTrPmwYcMGXTwvfqCr4bL
krk4UgGAwiAceeeRDfXABnUekmFfU1MTNm/3jM2bNnNmb0s93PcN
5C+Wj0bZIJBLo9Xqcc9hDN6oM0UJVbariFBcXsyqNYb9GnqGrEBGp9WUoCAQC
/vx8ajQZWqxVNTU1NTU3YtGkTbGmpaWPFIe6ujqo1WqmcBmQoIe6uticfj6O3tRVVGNgBO2Mo0XYG
Fkxfit7a1tS0Lx3W5r6eLuX+vGefrQlEfhS5WNLvMD+gDTvBqysrKUFWZhp/+9Kc4mM
dOoQ//vGPUSYPHKHGG1uPHG1dzMwtidHQUL730Ekv6f6eDc6rb1u5ao1eD6vVizxCT2dQP61vbR6YRYXdlyLHqZ2
RXs27cP6oTzxRI7MiJ8mKzqHWpsQkpjoqxJcQfpvP3n2vBgkhX3S0tLoo/8xfqoeuf4tRRakiqmRFqrhRpkPz5Y
dpcfbr/fsoJ6F8KUoQ6Nx0Zf+t0YIodD/LGt9TVpVcp2PYYfKS9xzpJ5rUe1lWw0TYwlVJmp21tOf2DeeJqKdhdiPy

Hk3OQk9kfGQ1OTEzAZrNhenoawWAQFRUVaG1t5egOihPwer0YHh7OkS0nEgn+7svKyjh+wmKxcMAl3ayPHDmCF15
4Aa+//jri8ThEIhFKS0vR0NDAg6CGhgbU1NTMmiG/02EymeZsbS8GIpEIJpPp1FfoJIhEIti1axf+8Ic/oLe3N4f
r5fP5TjrAIAivMxqNBpdeeikuu+wymEwmiMVieL1eTExMIBQKzYpmoXbM8PAw9u3bh3379sHpdPKEi27cRNC22Ww
nXR+TyYQdO3ZwG5CujZ2dnbjuuutQW1u74H0kJOYSafzCCy/Et7/9bRQVFcHlcmFgYICrEiMjIxgYGOCKJHEcKea
FBpUvvvgif0ZRUREbxK5btw7nnXcedu7ciXPPPZdvoORkPB/S6TTsdjvGx8chFotnDYDkcvmCBtbLcVPV6/WzbCs
oN7Gvrw/j4+Nslkp8senpafT19aG/vx99fX3o6+tjIv7JcOzYMTzzzDNz/l4kEqGpqQkNDQ0oKSmBw+HAfffdB4f
DgUQiwZXpTCaTY0xLRPmWlhY0NzejtLQ0Zx/K5XJs2LABGzZsyPm8gYEBPPXUU/jzn/+MV199ld2tHQ4HHA7HvNt
iMplw+eWX48orr8Rll102i7dIoIF6MBhEJpNBdXX1mogLOq2qtJNFfTz77LMIBoP42c9+hqeeego33HADt7Pe+97
34r777pvz/fNVaYlEAu3t7fjIRz6CSCQCmUyGSy65BN/85jfnfI/VIl/v2bPnpAM0ypqyWq2oqKjgpGixWMzE0W3
btuGCCy5YkzfzQkQ6Yf9aKpVyaywfZO4GnLgA02CHrBkW8/k6nQ4XXXQRt1gX+n+ESCSC8fFxOBwOeL1edt5WKBS
QSCTo6uri9ofZbEZZWdmCvGuWC6lUCj09PWwwGI/H0draipaWFuYv2Ww2DA8PIxKJQK/Xw2Qyobm5eUHHzXITIhe
KbDaL4eFh9PT0IBqNwu/349lnn8Xzzz9/0jaL1WplsnllZSUqKio44ZuIyZlMBjKZDMXFxdizZ88ple+z2SwmJyc
xPDyMhoYGllHTgL6npwdTU1OczO7xeOByuWCxWNDe3o4tW7Zgy5Yt894gTmX/53+HdO042TaRyaTH40F3dzcOHz6
MQ4cO4ciRIwiHwzyAGh0dZdNck8mESy+9FFu3bsXmzZtRX18Pk8k07+CGzu9CWKjz8nIep9lsFkNDQ/j1r3+N//
//2NR0GIglUqhVCqRSCSgUql4wtvZ2YnW1laEw2FMTU3xfiXLCFri8Tg8Hg/6+/sLVnIKYXx8HG+++eas1xUKBYx
GIyoqKtDY2IjW1lZceuml2LRpU06Vp7a2FjfffDNuvvlmjuyZz+7C4XCgv78fPT09PT09hLvd+L//+z/83//9H0QiETo
7O2EwGCCXy+FyuWC32+H3+2eOEbyuWCxWNDe3o4tW7Zgy5Yt894gTmX/53+HdO042TaRyaTH40F3dzcOHz6
MQ4cO4ciRIwiHwzyAGh0dZdNck8mESy+9FFu3bsXmzZtRX18Pk8k07+CGzu9CWKjz8nIep9lsFkNDQ/j1r3+N
//dM/4eGHH8ahQ4ewbt06vuj/4z/+Iz784Q9Tqn8apGve3p68Pjjj0OtVkOtVkMmkyGTyXB
5eT6ibT5kMhna29uxceNGdHZ2orKy8qQ3PIoUKRRFQjeMM42El8lk+EJOeUNisZjL5VqtNqc0ThJ7iusIhUJMZrf
b7dx2cbvdi14XMtczmUw5xFOVSsV5XTabDVNTU4jFYshkMigqKoJWq+WgXKGqS+iDQs7I5Dhrs9mWRHYWi8WoqKh
g8iWVxsnKn4ipsVgM5eXlaGhogFqt5nyuyclJ+Hw+aDSaHP8tg8GwaEVSKpXC1NQU2xiMjY1hcHBwTn8fs9mMbdu
2ob29navNZJBnMpnW5EThnYR0Og2/349oNIpAIIChoSH09PTgyJEjBUngwiw+mjzQUl5evqz+PrRO5ErudDrh8Xh
Y0KFUKqHVavla63a7OfONrsf53lISiYT5Q7FYDDMzM4jFYkgmk5DL5TzwFj4uR3Dy9PQ0RkZGOO4pm83yeUZO3LS
Qf5fb7ebr19TUFDwez5zvr9PpcM4552DLli3YuHHjkpWEqVQKvb29OHjwIA4ePDgnh1eIfKrEzp07cdttty3p8+f
CGUG+Xm1Vmkwmw5YtW/Daa6/x/91666146623CkpQgbVDvs5kMhyjIExNJwLd9PQ0pqam8NJLLxX0RCkvL4dEIin
ofkqlzJORpc1mM5qbm7Fp0yZs2bIFDQ0NMJlM3I8/WYlbuH2kuqKIgWg0imQyyYnZ9BoFCtKSSqU4goHiGHw+H/M
WpqamcngMJxtMUvgjBXMu9FTQaDQoLy9HaWlpDpcqkUggGo1y68/tdsNmsy1rcOZCIJPJWDqcSqWgUCj4Qgqc2G4
a2Pj9fthstkXxTBYLjUaTc5Og/UYZamQJMTMzg66uLvT39xec3UmlUnR2drLSbOvWrfjQhz6E9evXc3WxtXv6Wx0zfJ
+zt69e/HKK69g//7OHr0KOx2+0nPM6pklJeXw2g0ct0g3+/nDC+1Wg2tVotkMomZmRkoFAq0t7ejvr6eiz9+85v
fLJuKVCaT4YILLsDHvtTadkp7AcWOp3GI1G4XA44PF4MD4+jsHBQRw4cIDzi9Dv9So9Hgb//2b/G5z32
uYBV/MaD4HyLRm0wmFleQP5tUKkU6ncbw8DCOHTuG8vJybNmyhZ8oSmtubgYAPPnk09t37e6oSmtubgYAPPnkk
NlUrqGo0GDQ0NOY6b5eXlaG9vBwB85zvfwVe+8hXcf//9uOKKK/DII4/gyiuvxCOPPIIrr7wSjzzyCK688kp
jyTh0kybF3796CvxfO0snwjlqOmUyGBz0ikWhV5KfA2zwXyvVKpVIsXSePpnwUFxdzRYdIsfX19ayWamlpgVwux2O
PPYannnqKiZtVVVUoLy/H5OQkSkpK4PP5eB+43e4VGRzRRAAAZ0gBb2c0EYRS+5KSElRXV3NocE1NDXbu3MleSCU
lJYjH46xCDIfDrF4hv51kMskD7EIQ+mClUim2kujt7V3wtpF9g8FggNVqZd6X2WxGR0cH1q9fD4PBsCccja5
Znve/7LLLcNlll/Fr8/Fr8XgcMBsBsbw+joaA5XkKqCC+Vh5WP37t0FXyc+mF6vR319Paqrq6HVaqFSqTaQbbrgBEx
MTq3wEGo0
1Wo1169ahqakJOp0OcrkcHo8HHHo8oOo9SWsfJlMJvayEXraUDunqKgIXq8XNpsNYrEYarUaGo0GarUaer0eJ
jIyP4wQ9+gP/4j//Ad/4zne4Qdx88804//zzsWHDBpx//vm4+eabAQDPPvssrrnmGlx99dX4+te/jmQyCQCIx
OP4whwzyAGh0dZdNck8mESy+9FFu3bsXmzZtRX18Pk8k07+CGzu9CWKjz8nIep9lsFkNDQ/j1r3+N//3f/8X
GGzfiyiuvxCOPPIIrr7wSjzzyCK688kpjyTh0kybF3796CvxfO0snwjlqOmUyGBz0ikWhV5KfA2zwXyvVKpVIsXS
GTt37sSFF16Ic845B5dccgkuvfRSXHrppbj00ktx6aWX4tJLL8Wll16KSy+9FJdeeikuvfRSXHrppbj00ktx6aW
FxcU4+tnnMsNjc346abbkJVVRUPvsbHx9Hf34/t27fj+9/7Hr7yla/g/vvvxxVXXIFHHnkEV155JR555BFceeWVe
CQpGwo+7JOFsWiwVNTU1z/p7ImjRIpFTqQsTWjRs34itf+QrS6TT6+vpY1kwu6m63G5OTkxgcHMShQ4dYWQwGo3
4wAc+gAsvvBAbNzzA/v37sQAbNmzA/v37sXHjRgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDB
gDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgD
An+gAsvvBAbNmzA/v37sX37dlit1lnKs1QqhUOHDmF0QGj0YhsNouBgQGMjIx
gZmYGyWQS1dXVsFqtnBhOi9frRSgUgsvlwrFqtnaBhOi9frRSgUgsvlwrFjxzAyMoJgMMjfTyGQM7jwOygrK2PiK5mMkjO7MKyVDDYpYFOr1aK
iomLVSPSrDXLqDgaDkMvl0Gq1J70ZRaNRjIyMYGhoiPlxyWSSq5FarZaVqltFgqFgrl7lL6uVCqZ86dQKGAwGLB
161Zs3br1lLaHbqpzhSGfxdwQi8W44oorcMUVV3Cvrw//+c//jdtuuw1XXHEFHnnkEVx55ZV45JFHcOWVV+KRRx
Xs4pOeE5LJBJotVoJotVoolUq+n9NjbW0tzj333C3z+9/7Hr7yla/g/vvvxxVXXIFHHnkEV155JR555BFceeWVe
CQpGwo+7JOFsWiwVNTU1z/p7ImjRIpFTqQsTWjRs34itf+QrS6TT6+vpY1kwu6m63G5OTkxgcHMShQ4dYWQwGo
4wAc+gAsvvBAbNzzA/v37sQAbNmzA/v37sXHjRgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDBgDAvn37sGHDB
```
```

ewLFjx9Dd3c1cqUAggP7+/pyTTqVSYd26ddDr9UyIpos3DX6mp6dx4MCBeT9boVCgoqIip0X00EMP4aGHHlrUNlR
XV+ODH/wg/uqv/go7duyYNQBZSN+fql6nevzIZDLU1dXNev1kvKqmpqZ5B1zxeByHDx/m/Ws0GrFt2zYeECSTSUx
NTaG5uZnbZfnYuHEjNm7cOO960P9Rfls+qqqq0NnZOef/h0Ihdl+22WysiDl8+DC6u7uZt7YQabEQRF4ttB/JG4e
+9/xHjUaDyspKVFdXo62tDfX19Ugmk9zip3Do+vp6lJeX8/krJNEvBqlUigcaNIALBALo6+vDoUOHYLfbc0w0hhZ5
ClKQOgKuR+ZNQ4t+JRCKWy9NkiPhzywmZTAaNRsPXM/ouaCJBn03PhetksVjQ2tqK1tZWmM3mnAgiEjeQm7RMJuN
r4FK8doibSe7YK41MJoN4PI5AIAC/38/V7jfeeAPRaBRyuZyv49QtiEaj0eIFm83GlIZgMAiv14tUKsXXfWotTk5
OYmhoCMFgMEeAQzEiF01H2KxGPbs2YM9e/YseZs/+tGP4uc//zmAs6q0WVhOVdovf/lL3HPPPXjrrbegUChw8cU
XY+PGjXjggQfmfN+1FAnybgG1V1V0gVMteizFizyNXC4A4Hkskkq0jogkmzf5lMiNzXC4XHA4Hkskkq0jogkmzf5lMxjevcDiM/v5+vPnmmzh69CirU+R
yOTQaDVQqFaRSKRQKBcr1Ly3PUJVqt9h0VLfFOBt1IyCOFYnPcbjfGx8cxOTlE9dqJUGcKolEglAohFg
shmw2C5FIxDyZVCrFbiP2hwdBi1rmsrAxGoxFyuRxisRjxeLygeCKRSLPCPEn0dqJUGcKolEglAohFg
ppIqV0ICUPKlkMhk71RN/jyae+Zw9Uj5StSN/Eb4uVI6RBUIwGGQ+XqHJ8HLum+WGQqHgAVlJSQmrgSk7UQjiHhL
flLaROIvAieNPqVSitbUVX/3qV5d1XejjSjttFSPaEdPT0/iHf/iHguRr+pujR4/ihz/84UnJ15s2beL
OO7Bu3TpMTEzgpptuQk1NDUwmE4qLi5HJZFBTUzPvOn75y1/GP/7jP/a3/b80T4kzMFN2/0cUvl+MT9nK8jKoeoyxWbT
IHoEUnnv27MGVV14JZxMJsMRKvS/+Y9kWzA8PIze3l6MjYpezpMLomUIydhpAFMJ87WpVJozKai
pqWHPIFo3Wk+9Xo/S0Q0DeZTEIsFnOOWCgU4keqfFGiwEL391wJ8aFQCDMzMxyKSi1
imigJXarzbTVCoRAmJyfR19eH4eFhhEKhWV5DVBnJZDKIRqPwer1Ip9M8aF4qPNXC1qtFnq9HlqtFolEgis5lJs
5MzODaDQKiUSCyspK1NTUcHuaYp9pgNBpzuFo0QCsrK0NjYyMMBkNOe4uqnUajdkWkVL2bC2+++Sbuv/9+PPn
kk2y8OtfkQiSR6PIxQKobGxEZdeeumyq9Wn+WitOuSrvttttQVVWF+Hf7L4d8ffPNM+M73/kOM73/kOAOCSSy7BlVd
eicceeyyyHfH3XXXfhV7/6TN7/6Fb/nN7/5TVxyySVIJBK477478478MADD+Cmm25iA7LeVg6fTafzP//wPHnzoAQoh
aUqW9U/BO3z7g7DaeSciPedBoNBzqm799hT/y04iGAyywzQNhNRq9x3f5/FaP
JJKanp7F9+/Y5v8OletgsFoXKS5HKWlpaf8PtFolE0OLetjtM1mcMjbS4RsSr/sjJMpEo2DNBxbcQUhpXjdMsL2vu
22NgYotFoTutJaIki477BYGBxCdmR5NuSUG4l3afTEcwZVsGVE9e8jJgG4WDDHSYHjkfjPR5hghzHWFiKOTQXdhZhlpFYKFBMdOiNkkpPqfj5tpZkI
Lc1qiBoMBVVVVkEgkOH78OF5//XWUlpYqgKX0o8mt8VYsb+Qc1XEN4lOZn0z/2A/a/6H5bLYm/s84R6kJUJEaTi/C
V1WHbtm0YGxvLec+mpiZs374dk47PmgjPA3wH9hbhHlSRJBYf3h+ZT+2uT9iWUGZn/AV4WpvyR81fZRJ3Tm0Y3d4ix3zttbpxVdVVl+mxGLNc1+W6
Oi/pZvDtAyksCqS6NRiPa2tpO45qd+SguLsaGDRvQ3t6Ov/bfjyv3DWVx/9BZ+dZl5U2GUkLtz0LdLbMLtVD
hCPmJq+2uL8RpJV97PB6o1Wrs3LkX49UKoXnnsODz74IL797W3nnBzv+j94Xoj7Hg6bLtXdbbqQx/6EG699VY0NjbC4/H
giSeewWPj4OH75y18WfH8+Vqt6Vs86eOng2bJ2ly8pu6cpG+Gr6l4nGrp2bJeaaAB7wsPEDQe7VtRl+Pb5zLtJA1ZaB+G
VleWQQ08VVFL3+XycoF1SUsI+XE56nmmQEJmRE56nk6nmQdB7QOyNhg
bG8PIyAhGRFnbHWIRCLe9/Td1dTUoLmwPz2uHhwPVuEmq0QDmgcFk8r96gkmUuRFX1buegZTKZkxKaye2dhBd0DlL
LkfIQhcavZOFB1Zt3+34+i5XFaRsYDQ4OAgC+//3v9/n5JyYq7Vvf+lB9w6aJYq7Vvf+lbB9w6
Hw/jyl7/MyrN8UPZNRUXFvKP91VKlvfnmm8tmf65SqVBYLBZoNBrmVJGKy+fzIqNcs83mUwiHo9zj5r
IcCfLoALAGWGFlmQymeNYTQMhIhoSqZAGgFQ6pvIxuUrLZDJWyhEXQnVAWyhEXQiaTQSKRwOVyYXJyEj09
RSHjQSIofISYnJ/Hqq6/mvEbRG9XV1SgrK+NBmtFfwWWBgtSANRoWPer2lTzZbJYTAmXqHuWoUhrWJy
0OpFIJFiertfroVAOYLfbMTQ0hHA4HA4vFgu3bt3t6Oioi
EQiG43W6kUinI5XKULJSw2ojUYDQ4pufZbJb3GR1f1MIIi
7kEKJiKMGgwGtra2ora1FB5oNPpYDQaUVtbi6qqqKj4maHtTqRSrhsgpfGZmBtPT043jH43w
cxGIxDjE9ePAgq8Zo0JbJZNik0+/344EHVvUSiQRVVVs1FdTU8N5aT09u7uZJQyqVQnF
xMXQ6HfN66DNCoRcpqbmjHahqJuTGCKRCmcdNgyWg0oqWlBU1UN1NTewF9sILL+C5556DTHewF9sILL+C5556DTH
SkhJ+P1oPoQs/PRLpXSwWwmg0T9kqWMQqGATCZjR3xaTxJQJJNJOZ6SkayKRvshPMOLxeA7xnqBSHj
y29/+FmVlZZxqX1FRVwql8uFQCAAVjM1hXELVpuxON+x3+3nMn8+zXEGkSVjM7td0zaBzq7i4GG0N
eZaq01Y4EEeLmm2/GU089/Oc/5bTl0tVp3dzhbN17985ZTl0tVp3dzhbN17985bTl0tVp3dzhbN17985bTl0t
36sVALpejpqYGRqORBwfkhCmVSqFSqdMy1J/I0TfsSSaukfgaxWKx7WUhKsSkDc5C4vcM/Vdm06v//0TJDMI0aEByMUNfM9tpyRmIvBMu66tQUmXKQ2RHcVthkO
KK5tOBRaLBfX19aiqqoJer2d9NeVxyFXuRgWDxcxosNPN1qNUiuSKHAAFOSQHWM4AWEjbhcxU4Hj0LM5LVoCXgB
A0dFRxONxpNPXmcrSTopkJp40IIq8RnOmgQBKDgd0yyeiIMJxKJNX3unEkg3ppMJkNpaSnnNioUCoTDYRKWV
qHP5avuAgGo3yAL1QJZ0mCnq9HkqlgfF0C7lPXXzzxbj99uttPfWMFOCOy0nw+H4xGI+68804ELj+7nAzFJGxG
WjzzyyLyRIOR8TdBoNiwiyiEQi+MlPfYKvQmYa8ofPqyDFw8d3j4kfjePv0OCIjMhv1aH9nq8Bz4t6zfN
JFYNgMMhEz6KiIuh0OTolJvWyh8ViKRIBqN4uWXX8ZSiKRIBqN4uWXX8ZFF12E4uLinMEKVR
pIXksrTORRWmhAUgqlUJpaSmqqqrQ0NCAioqKiLS5ofYG2qVY3KxOhHCKcSqXg9/v
ZiFH4mP/azMwM0uk0stksKioq0Ev2xxPFoB5UjSxLQlmKIFRXEajYYaDDy9ePbpmb7harzZrnzK
ZjKt0arUaRqmUK54+v5/9gI0yOTz/9LSHBqMKyUVW5lVRaQ+q4iY6DAGKRxgKJCzQnRzZFBaB
kQSYpvcPhQG9vLyYmJmYRXeMd/Pc9n7GatVCGU7Gq5XxdVaEPQAuAWXoEtVhXhH75Z+8TNFKWtqq
c3Mw3SGrVy2QyNjil38XjcYyMjCN/ji0PVDKo3GIa8vjmq9PscGN0ZMnt2uyl3hFcLKbfjHJlHUXjaZ7Xsm
il98uo/0rtAAQhif39fVhZGSETV0EBKxwWcXVmKGQBryLlTk8XTL5OB4lfKJ3udPGxuKxFjwkeQUMJksP3
cHEqqRiYSCT4Ww+EwPB4B4ZGSC6XzDCSoCOSoVaw+UkEnSmV6XCCnKD8zuHd1+GazG6S6XC729vejt7eKKnDCSoC

qR1TRWSvneEVFBUQiEdxud47thEQigUqlwnXXXYcf/vCH776sNIPBgB07duDee+9Fc3PzLFXaZz/7WRw+fBgajQY
33ngjPvCBD8xSpd1xxx34/e9/z++ZHwny/ve/nyNBQqEQ7HY7B2paLJaTruOZpkqTSqXo6OhAR0fHMq9V4c9SqVQ
L/vtkMomhoaE5yZDvJMz3HVZXV6O6unqV1+jkmJmZwRtvvIHu7m6+wAaDQW790WIwGPDqq6/ygMRkMqG8vBzNzc2
oqqpi8zhqxxIHLhaLwe12c+YeLVRFo4HhOeecg3POOQd6vR5FRUXcelgM5pKFLwTzqX02bNiwoPdobW3FxRdfvKT
PHh4eRjwe54w/mpREIhH4fL6cAZFwG6lSuZDtJnfqtRQJIpfL5+0Oq+Li4gV5l5K3vB+EzblOWgulBKpzf+8l5ILSc/JwMpvNMJvN0Ol0PNGlqrzf74LSY
MQloFfc7pzEpbM/72Kx0J8uc//xmDg4N48skncfHFF8Pv98PhcECr1a66NDUfyWSSZ8Hj4+PcuqLZ6FksDZlMhme
s5FKr1+sX5XabTqcxNjaGwcFBDAwMYGBgADabDaFQiPkdZrOZ2b9R9fX6bT6VBaWoqGhgY0Nzcve5XxVECzSDL
l02q1uPzyy3H55ZfP+3/JZBJKpXLXLOG45SqVzQpGMlcaYSc6VSKVpaWgr+jq4H8/3vWZwFUQAKQSKR5ET4WCyWFYl
MIaI8kBveXggkVllreNdEgjz44IMAwDO5G2+8EQDwyCOP4P/9v/9X8D1Wi3z9i1/8Ys51oJIq+T5QCZ/KuXQTkEg
ksFqtHLHQ0NCAioooKnl0CJ9pfxBcJhUJMLKTyNJETiSgaj8dzyszRaBQajSYnQFCj0eQo5qRSKfeSqdVDHILx8XG
8/vrrPPAVLlKpFCUlJWwwJ2ytSaVSJpiSuZSZlMJoNYLOYZCZnqjYYmmGHW4/HMakEUFRXxDMdsNvNJSf3yaDTKBEh
a/+UqPxNJsq6ujknCtK2k/qOysrC1mU9MF7Y+6HdU2u/t7cXhw4cxcIixkUilSqRRGRkbQ29vvL34MQ2sQN/YABBBNw1t2G/6tbfHjQsH9+vl+h0YDcsI1
arcamTZuwZcsWtLW1oaGhAWVlZdBqtchms3A6nVzcbvdcDqdMJvNkMvlcDgcGBkZQVdXF8bGxmA0GmGxWCCTyZB
OpyGVSrkla7VaYbVamVA+1wWc1ov4JUSCjcViSKfTTMRPJBIIBoM8OyakgqgOyaqgn1xOBFlqDZZCwCwW45lsvu8MtUV
OJ9xuN6ampidSIRir9eLycllJdHV1wWg0QqlUcuaFvJyuqVHCNBjtxCpNNpTExMYGhoCBMTE2wUKJFI+LgUHrs
ikSgn1oJI6bQfqVpALTgAOce48Dn9TNfE0tJS1NbWoq6uDmq1Ouf7i0ajajcDqdmJ6eRiKRYJNDynFTq9WLvjnT9ZE
cr1cKVFFJJpOIxWLcQgwEAhgeHsbg4OCyRjctFaQkLCoqylFRF5qkUFyJRCKB2WxJnJLL8nKn4A7zLy9WpH
hEonmJX0TVot8vXfvXvzbv/0bgJUhWC7W8bnC7W5v+dBqou5g+SFgqpVIry8nKhCvKhwOIxA
IwG63Lyhx/t0GrVaL2Gw2GjLpaHiCGw2GMjY3lBNguBuTOWgmDmZSZTOW+gmLpxA0ERDyAcRXpTj8ThXcXGdmZpa0HtXsV1SgvL0C0cM0GmV
1GVXbKisrUVpayq8R/0mv18NkEsFsPtdmN6epqNN6epqVSRTMSteHaDTKpFeaNJHya3Bw8KRBn4sFsFtU5oIBEMBtesmEC
r1aK0tJRVenNZbAhBgyyqoJJtAE2UhCCFL4EGR/nGjvlQq9wsiJVGuXUpVKpOc0gFwKdToeqqirodDooFAokFAokk0n
4/X4EAgGeyCsUg5o5Jbd3oSqvuLiYxRRkbyDMDaj48/NPAmQY/xDMljlshqFQqmM1maDQaDaPBINBMYEBDt27MhJnVgOnFG
RIPOVvRc7Zjv//PNzVGmPPvrorEHRYrBakSCXX3457rrrLzwzo/mUix+o/mUxiYmICIyM
jGB4exvDwMAqUMEMIDb6yJiNiBzf5hwyBzeF2JJZHzaPCOu8GxPUdWFE9EoQdaTyuA/Oyk1hFDNHR+v+NiJ5Cza8oYj1XYEnexbcaqj...
rLxW7d+9e1c8D3vbGIF4F8bSoBbsQaLVabvPU1tay8ogUNJT953A4YLFYkEwm2YbPZWBm2b98+jrg5VZBRpzAiRaF
QQKFQwOv1YmpqCm63GzabDRKJ4SR4aEiGQm0arcW9f5FFhm1xPaUpSHRaUCtdBZHR87iDnRSQS8vBIXSvUzIk2BcKhdiS5YmDvXH7...
...

ckr4v4f7JZDLwer18zIjFYhgMBo6sEu5bs9mM1tZWAGdVabOw3Kq0Y8eO4bzzzkMsFkNJSQl+8Ytf4P3vf/+c77t
aqrSuri789re/5QMKQA7hNt8sT8ifIGJ1KpVasAIjH2KxGGq1Glqtlt1ri4qKIJfLuUVBfAgic9LgivrVAHIqPML
QR5Jd0w1E+Fz4SCcFtXCI/5FMJnNcaulzM5lMTqit0EaAnut0Ot6n6XSaVULhcDjnMZvNzrr40lJWVgaDwbAkfxx
yux4eHobT6WQbAXLAJaUIVQcpzFaYz0T7hW4kywWqzp0OUIWTHK1JBeRwOOByuZjnsdiLIjm7C6vK81VaAbBTczK
Z5Ew/r9d7ykrO4uJibuPmo5DMmxy+SWEmJC4XFxejpKQEYrGYHaCDwSAikQifK6FQqGCUyFmcRSFQZIfwWk1dgkQ
ikXOuEFGfzCHJVmUhE2sy5gyFQgs+Nk+3Ku20Z6X9/Oc/x8GDBwuSr/fv348777wTL774In74wx8uOBIEONE7/uI
Xv4h77rkHQOFBDsUlzGVAdaZFggAnLvLURvL7/XwwCy3thTdias+spBcHsHzbt5ax0ttI1RNS5lG1iryiotEoy3b
NZjPnmY2MjGB8fJzJ/ZWVlWw4mUwmYbfbsX//fhw4cAADAwMYHByEy+Vil1yFQpHjnzU1NcUD8JKSElitVnR0dKC
xsRGJRIIrq8KZPc1wh4eHMTY2lqPiWiiIRBoIBHiwQeKIlah8FhUVoaqqCpWVlVAoFIjFYshkMsyhEA6kKfaGKil
DQ0M5BGYi3weDwUVVKpcC4pyQwEU4kSJD0pGRETQ3NyObzXLrOv+R/HPC4TAcDge3qF0uF5RKJYfK1tXVwWq1Mjm
YWunki0ZVJKVSmTPREk6ahERfqn4SYTy/EiNcR1o8Hg+Gh4cxOTmJ6elpBINBaLVaNDU1wWg08udTFBJVef1+P7x
eL1KpFNM0hCTq+Z5ns9kcz6W5fK3o+JiZmYHNZmPeI4UDk1cZDXBpIWqC0MOJXisqKsL4+DgsFkuO4lWo8qJKKu2
rlRww03aQlYGQgJ0Pq9WK2tpaiMVi3hdEuqZ9fMUVV+Bf//Vf332RIHTS3nbbbaiqqppFvr755pvxne98BwBwySW
X4Morr5xFvr7rrrvwq1/9it/zm9/8Ji655BIkEgncd999eOCBB3DTTTexCqmjowPPP/88//1HPvIR/OAHP8BPf/r
Tgut4pkWCAOAWS1tb2zKu1fJhJffdaoIu5IXaQSu5jWT8uBjMF5QslUqZLPzRj34053eFtpHIyRdeeGGOKmwxSKf
TsNlsGB4extDQEGZmZpgc39DQwIHQwNsBxuS/AryttiS1Gr2nz+djVZrQmyjfHyZ/oUpNcXExwuEwXnvtNXzkIx9
ZcLuvEMLhMCYmJmAwGFBaWprzPsFgEHa7PSdiIZPJwG63Y2RkhLl8NIGRyWQIBoPw+/1c2VQqlTCZTNBqtTw4IB+
kk9lLnKqykKwY1rLD+FpQwa4k5to+mpiQLYvwuCPjWjpeIpFIzgSLIkHKy8vZ3FPoqUfHZDweZz4kiU7mOk+oCky
DEqF9ykK2ETiDVGnRaJTlewAwNjaG3//+92hvb8dll122oPegC4bP58OePXtymP0dHR247rrroFQqIRKJYDKZ8Mc
//pF/X1dXh23btmFsbCznPZuamrB9+3YAQENDQw75GjgxAywrK+O/J4+cs1gYMpnMoitLwtnqakAoi52cnMTk5CT
PWhQKBSwWC7dwzGYz99nz2y0kKZ+enkZ3dzeOHDmCw4cP48iRIxgbG+OTVqvV5ngmzczM4PHHH4dCoeDAzNbWVnR
0dMBsNq/pm0k+aAZdCESAXwokEgkT0CnkeTEgtWX+e9L3cCpIJpPwer2wWq2npMhTqVTMkcgH8W/y0dzcvKT9sdp
YixEOZ3ECMpksx6ZBCLFYzFUs4MQkuqKiYkXXR6FQoLa2dkU/YyWwpCP86quvxrXXXovPfvazmJ6exrZt2yCVSuH
xeHD//ffjc5/73Enfg8iqarUaO3fuxN13343169cjlUrhwB+PMm1vN9Rc6fJNxKLkiFxcX57hyCytzC0UkEoH
T6eR2jV6vh9FohEQiQSaT4X2Ub9Ao5FFSVEehReg3RdeI/OdarRbV1dVzViNoHxRSM5/FWeRjSQOjgwcP4vvf/z4
A4De/+Q0sFgsOHTqE3/72t/ja1762oIER3Ui+//3v49ChQ7uNUaQ8++CCreJ599ln86Ec/mqVK+9a3vlXwvcPhML7
85S9DIpHwDIxUK9PT01CpVIhEItBoNPP6SqyWKs3j8WD//v3L9n4rCbrhLhbC8u1KQyQSwWKxoLKyEhUVFSgpKYF
IJEI4HOasL5fLtSAn1OLiYjQ2NmL9+vW8NDY2ci4XWR24XC44HA4cOXIELS0tbPQ3NjaG3t5eDA8PI5lMskneWgb
d2MxmM4qLi1nNQ2RwuVyO//7v/wYAOBwODA4OLmqbVCoVysvL0dzcjA0bNqC6uprN3gYHBzE6OsoWEEKTxkILcKL
lrVarYTAYUFFRgfLy8ln5cgBybrY0UEgmk3A4HKweFIvFGB8fx/333w+bzXRsNm80Gh8OR0+Kiqm8hZRHNWqurq6H
T6RCPxzE0NISLLroINTU1XHWkfK1kMpkzYSgqKuL1IEmzy+XC6OgoHA5HjjO6Xq/H9PQ0ent74XA4ent7
TbEJLkxRqPc/MzHC451wDeaPRiLKyMtTU1KCtrQ2VlZWzBhz0HT333HPYsWMHRCJRjtcZzdjJHoMG8fk/6/V61NT
UsDJNOHFNpVLweDxIp9M5lTvhxG+5QRWzyclJ+Hw+eL1e9Pf34+KLL0ZpaSm2pZSm0Wi3bYcRiMTidTthsNkQiESiZy
8R5pozDVQEw7YyO4jP85JJBLl2JrM5SFH/5NIJBCJRApK6Av9LJfLMTg4CKvViunpaQwPD2Nubo4+XC6OgoHA5Hj
+oOWOGZpoL2bCQueNSqViLzW9Xo+SkhKW8As9l+iYJV5xbW0tGhoa+Duj7EOyjVGr1ejo6MD73ve+Mysr7Wtf+xp
uuOEGfOELX8D73vc+rFu3DgEGfOELX8D73vc+rFu3DgEGfOELX8D73vc+zn/atWsXNm3atKD3WO1IkLfeeitnFHzw4EFcddddd
LlNq9VLWPUDV5ySWXnMqqrgjmUzMtpEpMGYf19fUFf2+xWNDS0rIs67pUJJNJjmURbqNMJmNzzlOBVCrNmZSuFio
rK1mBSd/jjh07Ch6n80W2nAk4FdUd2SDQAI2qRKFQiMUkxJPLX1ba1kWIM06V9td//de48MILYbfbcYbfbcwIq3/e+9y0

5B2ilI0HuvfdefPCDH0R1dTVcLhduueUWiEQifP3rX1/S+p7FWZzFWRDmC5gl0Ez+VG8usVgMHo+HXZzLyspQWlq
KdDrNbdHe3l7Y7fZZrsnUgnW5XNi/fz+TpIWEZpVKlRNLQ07jhUjl6XQaTqcTbrebqy9ms5mDSAvtI2FrjFr0Zxq
oBZ1/fxIm3tMj/Y1KpeJAYWHFUywWQ6vVLmo/nEorlHLzFAoFDAbDkt7jnY4l6y5pACLEYgIhVzsSZHJyEtdffz0
8Hg9MJhM8Hg9uvfXWeaWEq0W+/tWvfoVPfvKTs16nQWKhfjw9p5G8TCZDRUUF6urquLxdVlaWQ4QjewIi0uZb9Hu
9Xvj9fmQymRxYLqYk+FdXV0dh7FSzES+hb3wkeTVAwMDqKmpgclkWjAP6FSQSCRgs9kQDAbZkbu2thYqlWrR72W
321muPzAwAIfDAZ/PxwaZVqsVlZWVsNlscLvdfMEvKiqCwWCAyWRiVVghXlZ+knW+f0i+Em0hkml63+W88dBxX+j
4T6VSyyLlzmaziEaj8Pl8bMKXz5cjzpLH42ESrFarRXRXl5OUwmE+9/Wsg5XNhWiMViiEQimJiYwPDwMN/obDYb/u3
f/o0/n4wu55pBU3tEIpGgpKQEBoMBNTU1sFgs8Hq9cDgc7HpeVFSEyspKWCyWnDgOiUTCPkQAOEKEjDaJC0Ktznw
sxcX86aefXvR3IyQbk8Ltvi-VIwDxWIxdDodD7ryncnzn+f/TL48FNpsNpuhUqkglUrR39+PP/zhDznigXA4zPYd1G6
i2JJ8NWz+Qg7pPp9v2bly1NIlR+l8bmmhlACq+ul0OpSXl8NqtUKhUORcQ4RB4jTApcdCz9eSFcN815pTeb+FYMH
k62uvvXbBb/q73/3upH9zOiJBCE888QRQuuOEGjI+Pz+vjsFrk61deeQX//u//vmzvd6aA8oLIdZZI36SgSKfTTKg
UKowos0p4kaSZGvXxQ6FQwRkdcMJ7yGw2c8m4uLiYSd5CCwQkKkR0fH2d327UCqVSao6IhTgrNZMmCIZ1OQ6/Xo7S
0lAei4XAYMzMzfIMoKiqCyWSC2WxGYZzGYz544ARvdDofDAYfDgZmZGChkMuYi5RNmaaHBvM1Q6GzQOGRCvkNQzknvsluEuZ
jgdFoZDVOLBZjbtNZIu38kEgkfNwGg8Gc32m1WlRWVrKZXn62GE2QSAyQT2qmgaOQGzPfoItuvuTe/E4l2S8ENEk
V8qpoYkIxLmv52CZxjlBJJ1TRFXL1pkehG7wwbJz4ggdDYH87n83HwuFDsQ8cgcd+WEytCvl6O3rMQCyk9L7ZUeP7
55+eo0h599NFZgyIA+NnPfoYrr7zypOZWq0W+fu9734tbbbrkFL730Ei666KKcbC+6WZMNfv5Cssp4PI6JiQmM9Rk
VkQABAABJREFUjIxwSJ/X651VvRHmn5GUlE6C0tJSpkSEjtpBhKLxTA+Po7h4WGWgpLSTXgxoAoVPU8mk/D5fL
b7TzLBDDLLr6QffypQi6XQ6vVsnsw+Q6Rjf5CIRaL0dzcjI0bN6Kx+Px+PyEyEuT5M9vtf
rhdvt5lZK/gx4vpmxMEQ1FAoxT46qlvk3xUJYiMVCMBjE8PDwgvfHXKDq4cn+BjhxVkr0bk/vT4TYTCbDJ4XCxsTG
4XC6UlpaivLwcFosFFFosFiUQC4+PjcLvdOec0Gky73W4oKBmHwqzCs5Snt5BtY/bBGKQDd7PZzG2yWCzGr3qW4oF
ggRQtNJHRaDSwWCyzWmyBQCDnfBaS6IG3q9BUGSQlW37F9GTPJRIpFIpx5ZQSy8ScsDhcKCzs5NtB4SZj8kcrz
oFEIgEMipCOaT82lRKpU5RPrFtr8IVGGj+1s6nebKYiKRYCqJsL2Znx+ZzWaxZ88ebNq0iaX75OEmvH6k0+kcrzr
hpEn4OgmXqEvi9/sXvV1zQSaTQVa1WI5lMIHaLLWhgeP311+Phhx9e++TrRx55BC0tLbMSr5GRY/DGGkeqrD+2hc
wLgSK/BD6jdhsNnzyk5/EjzbcV4KgXz/vnt/ftwxuuQ/86N+89N/88iGsXm/0yfr88wRxy86hdb8fJnH21wvR2mqmv
G+vXrT1olKXFA8U/rCMCGgG+QfwW9KjKcm6pPbqskfusxp9vMWmvhLZnRLjXUV6oASWZeIXsYCJCv7s79c87P4nRx9
8ZDIZ3q4q4zhnwnOATVMyiGfuIspSiQaSXnXJSwSaHBBmJXVOBPsq9kKJpGSJDZxSpz5t79rWh9wOI/2Sd5O/sBX
KnqlyGtxvgmn0wmDwcnCzdaGEWqvV4KgEkgMexW4cmQiCcRDab5WgBajOSCZ/wud/v5+i
RhSAej89qB9BnjIyMLGp9RSIRSktL2cG+paUF69+wua0ejajtLQU69atQyVvAqjQahcFggNlsRkt
LC7Zt24zaamhpuFQu5fyKRCI8//1/fo9Nd5J83+w2MkqZkPmk9l/QQX6ptcnadQIh4s8q8FkOdCP3fe1YdQhafXPv9Hn17kPhOfA1yxGsXa7Kcw6+HKaHIUE1tEHhaPFHI1zqMd
LC7Zt24aamhpuFQu5fyKRCI8//1/jh8Ph+OHz8Ol8sFl8vF/I94PI7u7m50d3fP+dlisRilpaUcHWM0GjlImX5ubm5
GR0fHablOUSTPz3/+c/T09KCvrw/j4+PweDwIBAIs8iDiIS0iiL3GSFnpcUyBrLBbD8ePHcfz48SWtp1QhQ5hdVqxejoKF5
99VW8+uqr+OIXv4ziyjsLmj9bW1LqKpqQlNTUsOymMUl5GSg0KHmlKD1FUfjMdZIHnLMwWtPk9D
IV199FS0tLVCr1Thy5AmWIC3HWcSyAVBFNUnRDpp6T2Tp6iKAR6bFbXxlvFkm8eTcRFdwPmdLkxpQVd63H/TNzpiQeTDDUXCRT5lmQSchaOpHrkhMdfzc9nXUFJzT1dg3EvTZa0fYVUW0/N2
ac889hx/96EeL/j+S8BL732w2o7y8Pj8HV1dW+wRwXdTPLDHQBALxeL6LRKJRKZU5mEsnHC7UTiENCCo0z8aQ
8GajaQJLY/EiP+V6LxWI5UnBy6vX7/UwVqqKurQ319p9Perqew6pbdaf1MQyqVqtPphNPpRCwWwUNQygUQm9xL7q6uuB
0Opf8vmRqSBE5xcXF0Gq1aKlpQWtAK1QqFqQFfOwiItGXjjWsFwFOWyanpxGLxvVBbW4v6+npcTh+ibuwmWzbv4pMSbLcyfepcPtjeiAO3HhRwuopAOjJFNSbZFEf9LaB0BwOapR6hJQ4EcT
k+lhUVISGhgZZWYBLhnAY+LS0tKC8cvX3YOVjqdhsfjgc1mg91uh8PuThuh91uh91u51xBu4pqEQqEeFv59SqURjC5T
I1Pm05vlIJpMcSUTNDD1SGyadTsPsPn88CyZQKxSqaBWq1kVQ9vr9/L5
4+DiSyWTOILG0tDQnnbuu0nq0wqlrWg5BWzTG6JnNmD8I2ZfwT33TXVzKuL6aodBvTphoz7xmfkuOF3mPnUWAfGtwcI9mmMkPlWf
v+HDdxajSljQWMhgM2Lt3L9rb23SheqU4Hd4cANPvEzJ7NmzB0VFVRmf0rWXTnxIu993vxVX3/1Vxxz0NP

TwwdfT09PwdJloYpRVVUVPB7Pst6odu/ejU9/+tNQq9Xo7OyExWKBWq1GNpuF3W7H5OQkJicnF3xRJdBFqLS0lI3
dhARm4c9LPRGJUCs8qfJH5nq9HuvWrYNWq8VVV12FTZs2MfmXvJCE0QrAiRvh6OgohoaGkM1muXImVOMsBvF4HOl
0+pQGtJlMBm+88QZ++ctf4je/+Q08Hs+S32uxqK6uRnV1NcxmMywWC8xmMyorK9HZ2YnW1lbI5XJkMhmMjIygp6c
Hk5OTfKxbLBZUVFSgtbUVbW1tpySDXenYk3A4jL6+PvT29uYsQ0ND88746uvr0dDQwIoe8tuhwbzwOXke1dXVwWq
15rSEVnr7stksPB4PnE4npFIpkskkuru70dXVBYlEAoPBgIaGBuzYsWNJflsLwaluYzKZhMfjgdvt5kda6Gen04m
jR49y3EMhUJZfKBTiyp1IJOKqk0KhgFgsRjAYZD+nxaCoqAjnnnnsuzj33XLS2tqKuro7pFhQDQtfF4uJiVg4ut0x
8sfD5fBgdHYVCoUA2m8XIyAgGBgbQ39+PgYEBDsQNBoMoLi7OsVqgwRfFsVBQ8slgNBpx/vnn48ILL8SFF16IjRs
3nnZPo+U+FwOBAIxG48oNjD760Y9Cq9XioYceglqtxtGjR2EymXD11Vejurp62RVejurp62RVsC8Ho6CguuOACACdI2YcOHeJ
4kYceegg/+tGPcNll0GlUuErX/kKPvjBD0Imk2HHjh24+eab8dWvfvWknxEIBKDVahe0YxeLhZJahT4jNHMmme/
IyAj6+vrwxhtv4PDhw0t2RqWs8FgQFVVFW+z3+/nZTHSx4VALBbDDYDCgpKSEZ4WF1l+hUMBqpVAp
9fX3o6urC8ePH0d/fj3Q6jaqqKrS1tWHr1q0477NLSgqqqKojF4lkVHOGjy+XCrl27cloc5AYslG9TgKpQ2p0
/6BSLxSgrK+OBq9Vq5ecKhQKpVApTU1M4dOgQDh48iKGhoWXbzxKJhOXOx2pt0a77eoz7nbWmGO4kmORu9Zh/H+eefP2eVMRgM4tixYzh
+/DgOHjyIdevWIZFI8PEwPT2NaDSDSK6upqNDY2Mj9FqVTmGDXmPxe+ZrfbMT4+Pue6K5VKNDQ0sMBj06ZNuPjii7F
9+/ZlsxN5N5DLV2sbU6kUjh49iiNHjqC/vx99+9fX3o7+/H4OBgzoRzMaAJL533dP5oNBr2Q6uoqEBZWRnsdjuuuea
ad+T3ON93KLTCyWQysNls6Onpwd69e69e7F37154PB7uDlCLOP/7UKlUOO+887BdN5/7AEcWuV/VvtvEJz/7oM49
7UKlUOO+887Bjxw7s3LkTW7du5Qrfamx0fqlfGGgrQZKSLlhtHE5z
RZVjxsMbw/r91BlotVxxzOb08uhttGCEK33+Fu+tR24feWhrMOZVvyeUHzx8DU2kG40R9+AKKXOYzwyZVtv
XNCT8/ve/j507d07d6K9vR2AwYGnTIEuNnY45pHb45CuytPl8XjwKfXH++m/5WNQ8+a3F8Z6bqF10cmGHs3jnF+s+nmyZo4CO5cHaIDgcnU5VkZ+sWHk11qz
Ik3AeD7KH4yFVkhlgZ+Y/fLQJq22LeEOqEd6Jb38nBn2hN49+JTgFLHn4EqtkkEkt89vcGzxUvWPjs+/IaGlLkdD2C3Ob/LxnR8DOJlv+H8lsAMeJYUlqDg/X8Ih6MwJTNkv25+mO0nNGV4MJiIfiOxVDc1k2Pcrcd+MKsgH8aOFaSeVRHjt8qaLhE9kxjrxGV3qrQpK0i6HsmX0MXRj9e8+FsLv0iJSzpsOAyqOD2ABzNi6+Xt
wqAGm+kGAm2oh5q2kq0vZ+JxV4rFU5zBUoePt5U8/7Vw5m9gr6OEnf3s+2JpJbNLoEQq3Ra9ovJuR5Kih+wlbW67OK8mz/6+2hqo5Zd6X3GL7+b+LBc5AWdPQQVB4Lr5ZUuKSwjZjYhj8m+63jUSFQ5B3yhaW+YvEHbP8EKO4nxrDGO6XnF65Kj6G4kYPzqr+RH99b1K3ZZRcqx+vLz4L6X+kkJhYTXtdv9vi4nZ3lRVLrl5t5Fw2p7LeG8DrRbDxTmSCuk+LChUd
IyAj6+vrwxhtvJUy8Vmd8rcLCBtjRBT3Y+T

NG2EymTi8cjVbwyutSlsLWI1t9Pl8GBwcZCVVf38/JiYm2A6E2kvCSQU9F3IP80ED5PyF8uIonBQ4wflqa2tDc3M
zmpqaUF1dDaPRyNcTYYWcBtsA2AGaFhp0ny6kUik4HA6EQiFEo1FIpVIYDAa89dZb2LFjB5LJJJxOJ+x2O7eb8/d
xOBzG0NAQBgYG2E6C7EFOBjJMpuo03feVSmVOq1AkEnF8ElFTnnrqKTz99NPo7e1lHlJlZSVqa2s5kog6EVKplPM
sW1pa8P73v/+0VYyWdMX51re+hXvuuQd33HHHklZwuWEwGLBjxw7ce++9aG5uxn333Yf169ezKu2zn/0s+vv7UVF
Rgb6+PjzwwAM8mLPZbNixY0fB912trLTlfG+6Sc2V/3a6kEwmmIZVKT7vap6OjY8n/GwgE8Prrr2NkZASRSATJZBK
lpaUwmmUyQSqWIxWI8KVAqlaitrUVNTc0pqzmkUmlOrEdDQwO2bt264P8vLS3FRRddhIsuuuikf+twHD48GFMTU3
B4/FweG5LSwvq6+shEone8aotYGXP8bWCldxGiuiSUngxIIk5tQip5SqXy09KvKZKXKSQSwYsvoirrrqjP8epVI
p54ES6HpKAwrirC0GRHuw2Ww8UfT5fJiZmeEJ9tjYGKLRKDt4LwZqtRobN27EOeecg2uuuQahUAherxejo6N46aW
X+LuldSB/LJFIhOuvvvx5XX301b/8ZkZXm9/vx4Q9/eCn/uiog7gQlFBNn5Utf+hLuuusuyGQy/PnPPf8TTz+Nqak
pfPrTnz6t60s3s379iEUCvF6FxcXz6pg0MVhtL0o4oP1G1rfWFJWCQPYRnZnJUWiQ4Ry1ZxKKREZT1kbhaqtixVNi8Wi1FfX4/Ozk50dHSgo6MDDQ0NMBg
MUKlU7DsiTC6nda2uruZQYaFBpEajajgclkKKlhJyWazmJqaYmm03+9HRUUFqqurUVVhcrKSjbBSyQNHLJBCmY
cccUVc27PWuBbrBYCgQBsNhucTidnPJWUlDDBP38hs1WL+p/o2koeesSVBaRCjvFFDrSaFQK1WL+p/iWQ/n1HkWZwARXZQVM2
1114762/o2koeesSVs9vtiMfjOdfuZDKJrq4uHD58GMePH0cwGMQrr7yCV155ZVHrlc1c1mT/t3t6SB0Yc//GHs2rU
Ln/3sZ5d7fZYEn8+Hl19+GXfccQeCweCseJGf/vSnAIC///3raGhoKPi+qY3HexBy7thJJMPnc7ZmyGy4AvYn4NzGr6OhMqr4Sy8GYMvxnPNGR4pdh8BTr
zz+Bv//ZvF/U/Qldl4G1jtLKyMpSVlbG0sry8HHq9o3MhDd98kGi2AB6FIvFLEumkiuRYaVSKYxGI8v6891qSXF
xqkROsVhcsK9ONyhyBA4Ggydt7S0nqqur0dLSwoM7agumjo4iEolgcHAQg4OD+MMf/rB
s6yEWi5TQ2RLsViMycnJJTnPm0wm5k91dHSgrq4YLNZuMB7/j4ONRqNSu8X8h+
JT0Et27KyMtTX13OwPx4LXXXsPBgwcRjFBWUxMqddDqNSCTCXBjghHLObbDZz0Ktw4JJOpyESidgHiQpB
DTqcTk5OT8Pv9CIfD6O03txec///k5c8YWAvKJIo8xWnez2YYqN1uh9frQ7aqk7AyVyjhklRnnzUgkYu
MXBW6Ngmf02K32zE8PAfyfz8fXCbpWkIN3LBbDvffey7YawnMidiM1WBLYL0uk0pqen+bOMpqa4aNWq9
aWspu0WVlZSySiMVi3PYVcu1oHQtABHFV1dayEE8ZJhMNhcjzkqHQqOzNZCCHCHQqzNZnPBidZcCIVCGB
b48ka8fFqampyQlEXAyLt00BwIVYE2WyWuUgOh4M5XisFmpQvBslVa6HQ6VFa7xOa7
2FiUlJchkMpDL5WxETCkMpO6j91oOrHgkyHe+8xcf//9uOqq7Bu3bpZ7JpO711sW95ShCq0q655po5/+4zn/k
M0uk0Hn74YQAnBhPzEbZXi3x96NAhPProozkavFvJUCcEjJcJOWX8Pl8fAIEIjcQcQj
Iv0W4ED+HVCfUY89ms0zwpYEGaeTlrg0i+GWUeVMGCo5MTHByph4PM77L31MTHByph4PM77iLg31MdPp9Nwu91MvBR6IcVisXmPBYl
Ewm7XarWaOT9ut5sH+PsdEIul/MxsdZASShUKoXp6EomCN3KxWMxiiiTPtPH0nQaPRsMf09nE7nvPY
GRUVFPLCg814mk7HSlYQu4XB43ske/Q8tEomEr11EwqaFXKtpER4vFPNEx1M6neb7AgCeDORfC5VKbfGhSO1
cXMxtMuG6y+VyKJVFqSnOuNSqXiCQx1BoT2FnK5HBKJhCfXQjWpAgUi8XYL4/Uy2azGaL1KfGhSO1
MyjThUltbu6bu6DW/2Kw4uTr/H5nzhuKRBgeHl7sW54SFqJK++Mf/4h+qd/wqFDh3hWcbKB0VoiX9PMgkzPaIBBGUM
kH3Y4HDzDp8fp6WmepdCqHhiqVCn6/nyVxqgVmEuWBpPdfAhbG9ia4HfiiKRCBaLBVarNWcxexeanwVcWNdf20cxeqBi
iGb5Go8lp3cViMTgcDrjbv4bSj1fC1b+iyW1LrVkTIMmn8/+H+4tm6+Xl5aitrZ3zGJqenoZUKs3Zr1QlVPDrq4
ujI+Pw+l0Bi7lZaWYnBwkC+YVqsVlZWaGtrQ11dHdHaHato5nK3JlPQSCSCyclJzMzM5KxjW1sbO1lTy5ja4iK
RCMlkEp0Tk7Db7bwcHBiwAeGB1LbW1tBG+wtqi20EdCk2Jp1RmQVhjqo0ECg2Oaf7+kbRJRkV1tbOb8w3lq5dBxlqs2yU6eMx0dXfkl8EwZmqhQuC3aBaZFu879fl
8GB8fh8vlYg6aRCJBKpXqwsMQEurq6YLPZIJVKIRKJEAqFMD09zdW4TCYDq9UKg8HAOVvkvp7JZj7JZGYN3IU/y2QymEw
mNDQ0wGQy5VhRUDVIoVBg3759aGxsZI8n4/4GvqSpMGKqzIBDuNtQwaZgTCDzENcUndaQRWLHewm4ZmtNyYoggIBFmd6A2uGbYOZ3Oee00dDodTCYT708Kk10sdDodLBYsTExpzp6sJBzZjTHs
4OEHe4/FwJA/dnImAT5Vh4UQpf6H1y88gFGbYOZ3Oee00dDodTCYT708Kk10sdDodLBYLH0sTExPzxp6sJBZjTHs
6QVXaxVhwfPSjH8XPf/7zM4t8vdg06ZVGU1MTRCIR9u3bh927dxdUpf3lL3/0B4O5/DgrJ71ddddh/e85z148cUXWXxEXs
vWiNfnywUtpAXQ3JDr9cXzJVbKObavoVwMqRSKdRq9aps56lgpYm7UqkUVVVVqKq/WvT/ms3mWa/pdDps74d27d
vP+n/L6fpGg0iKO28pKTktIfl0va95z3vWfL2UTu773VFbxlEH2Hu80Av3MzAxGR0cxMjKC8fFxTE1MjKC8fFx
N4brrrkNTU1PBVhBZnDidTgTgSDQW7vkbKLLBx0Oh20Wi3MZjN0O0l3Oe5DvEU1a8xV2hV6TSCSsdistLY
8yqKiIl4n4p6SuuvFF1/E5Zd/hdjSOBJmS0amIYXhpiWVigo3GazvFw8HmYzWz9kc
ymeQJCLUnJRIJ7xutVsteZZZYFAgNdndnfHwc8Xg8ZSthZwJnrR2NgIpVK55Pu6dwNDVyRHD
/MDg4iL/6q79CQ0ZhbrrpJqxbt2/R7rfs9upUm8GzQ/xm1V6RzXcVS0cGwQKerGHR+QqEKd0nfDO8aHkSoh8vMJZKL/7s9ETjtf3n1IV/S7yS2F1CD2Vr/Ow0DJrZfkFoIGaYGp6q7t3V3T2LJgOkF
YKojbaTQal/wehMU4hRNfRiKRQKPRoK2t7ZTXnfYrcKFxakHHfz6OdmdX5OfS85jbyLGgYUFpcpnRZzx/v6fyjhKSH/wVHdaWGWJ9rXnu2MtuR3Fx9LzPbBzXr5/gsmHgFtjs1sbjbbpalbF4B1q
NHl944YVlWbmlopAq7Xj3b7E3jtnRdsWEUXYZsGg6+vYE35wpd85O7nU05K4PPPL1yCzdFL1u9LKJWnVuMTFjDBhZcQsLYY6GDuNVbOBuEwedAAMdjg3tNAE1VzNdoD4+vmPfMs8SZ
gEEA0GkUikYBUKuXg1ELu3dRCpDYwahBgZw/PhxHD9+nNtMoVAI1m0Ao8ugC+D9WD0+yM6nhE0ybdj8AN1MsljtCvJ7zKnumYJqDwmkUyLxGsN7gG9qu+TzdoqKitb8Negs3gYFZs9VQV2rWNLA6LbbbsOjz6q666Cp2dna
N2Lp160krfmexNkEXaLrBxuNxPsbIUToajUjUIsFkOhUMBgCuri5HNUicD7lcflqflRepzZAXtrkRKkZqpXJFryRNQhYDgdrjHl8PxF36uxNVj6VNJNOMFdtDGAXRCnC4NeoN6S2ZjoV1OoF6cb7LDZKFs4OJeMbIcC3qXFUpk
xTWKxGLfn54qpSKVSXBkh53XMBLDtsmZ7qpsaAVBE6HDWU9qyzNMiXPtlJ8uR3wyK1ob8jNgcQMoNZwH4GnXFj2LWa9a1bbWzKK6s4K9QHDWBnu4jbR9GEgzAc
9QF2oKq0QyMCwEFZLlfaLX/wCN95446L+RywW5ywyVFLBbCbDYzp4Bu8sQRootYLBZjszN6FL5OxoqcBL9ZeHIW6VyeUQ
BQADhcDiHk0Tp8tRyVCgUkMlkiEajOc656XQacxfCwJFX/lDiJjkeNtzqSjyKSBnIe0Rff8Mwn
uTExMLHgfUjYV8SpCodCie/bRaHReU0Qhqqur0dbWhvr6epSVlcFgMLC7LcWWRKNRJlwScTE/QkGr1aK8vJvzz0+j
GIRKJMD4+jv7+foyPj7M6kLytNBoNFAoF3G43R62oVCqUlpbi3HPxcaNG096Q5xlKRKJRGC321mVtpw3fxqwUru
itLSUQ0fJbVgsFkOpVMRFt09zFIzP58ovL76I5557jg3xRkZGGWJJ1VaNsLQSaTobq6vGF4HIFAgEm

2lJVWWVnJvB6NRgOdTger1Yq6ujoUFxczYZ5ajbRfFQoFK9P8fj8ikQhfA2gfTUxMYHBwEH6/n89/Mk51u90LWn+
VSsWkeIlEgoqKClRVVXGqfSAQYAPZ0dHRWVwx4ISPDeWHCdWwc30+DXzyk97zQRyjoqIiNpMtLy/H5s2bsWHDBtT
U1MBgMODgwYMcpG2z2RAMBjnaSGj8ajKZOBqDFHlkGkntLSIeC9tdyWSSlXeUFUg8r5W8J0ajUUxMTLCRL9lInG4
sdOJMIcLUQhOLxRxZRfceuVzOleQzRpVmNBrxP//zP3j/+9+/2H9dESxUlSaE1+vF5s2b8clPfhLf+ta3Cv7Naqn
SXnvtNXzve9/LOalIkUAX87NYHMjpW5jNRDPRmZmZBSuHJBIJzGYzqqureSHXV1KuUExCMBjkQQ15BTkcDgwMDJy
S9Hu1QAnaFouFowEA5PgrhUIhjgwAwKZt+SGTNFgTDtwp9JgUPDqdDmazGcXFxYjFYuyWPDo6yt8PDejzvy8a7Au
VbCKRiKXF6XR6loydziOqJJ+qGo/I28IqEFlcnG6l33ygivpKqOdIdfVuh/DYp0GcQqHgiSQJTui8okGFkAQv7Hw
Ab58LhY4tmiyRHQaJdWiiRaR9hULBUTT5E06avPr9fuZCCS1hSH0mkUhYTi+szguz+Oj8J7WcSqVi65f886MQifz
iiy/G7bffvqzfyYqr0ioqKvDiiy+eUnLuckKoStu+fTvuueeeggTs973vfQBOXOgvu+wy6PV6/PGPf5xzlrlaqjS
azT3//PMFGfhC5ZaQ0Cf8ORaLwW63Y3JyEhMTE5icnITX681J55bJZNDpdCxppxK20GY/33KfLv4kaaXqEZ0M4XC
Y9xNVmWjWRYtYLEYoFMJbb72FTZs2MalQ6FOSSqVYVkonIJlcZjKZnCBMoQxVo9GgrKwMNTU1aG9vR2NjI/R6/bx
97Ewmw5UzkuuTgWRxcXGO2eRiZn7zqdJ8Ph96enrY74dUNHSxohksJWXHYjFueeTb+09PT8Nms8Hv9/PgmQYJFRU
VaGlpQW1tLSoqKjhYdmxsjC+UBoOBLfnD4TAmJibw5ptvzgqmXCyKi4uXPYg0H3P5W50KSJJPVdBMJoPt27ejra2
NA1DppkFRKCUlJXNWmjOZDMbGxjA+Pg6VSsXxOBqNBtPT0xgYGGAn4XQ6zTEq4+PjGBkZQSKR4CBe8n6iv4nH41x
50uv1XFlNJpOQy+VQqVSoqKhAY2MjjEYjVzvohkWk3z179mDHjh3IZDJ88yOlVygU4psjDSCnpqb42mKz2aBWqzm
oura2FlVVVVCpVDz7p0oc3QSFlWBSpZGwRVipTiQSvL9Isp4/SRQe8zQxGRoawqFDh9DV1YWJiQk4nU5oNBps2rQ
J9fX1PGigygQFxFI0EVXE8+Xtwpgk+pleE4vFmJqawtjY2JK8w04FdG0mmsA7CQqFAh/+8Ifxk5/85MwKkf33f/9
3DA8P44c//OFpb6MBb2elffGLX8Rjjz0GnU6Hf/3Xf80hYD/00EPo7e1FMBjE5ZdfDqVSiT//+c+LMuYKBALQarU
L2rGLxXKqfdYiTsf2UcmWBrcymYxl4Cf7P5Is+3w+dHV14ciRIxgcHORBjbCtYzabYbFYYDKZMDU1BYPBgGw2y4Z
9TU1NaG1tRW1t7armfi0G2WwWQ0NDGBwcxPDwMDweD89khS0HjUaD3bt3o7KyEkVFRTAYDLBYLGhubobRaEQmk+G
kbOFgORQKcRo9vW632zlehWwXNm/ejM2bN0OlUrH5Gw1E6MZLg1qqXNEMfGZmBmNjY3A4HLOsKWiRSCRcsicndeH
x+E4/D4Gz27gQLJZPmclkcsw2hVQEGsQFg0H4fD5uf1K7nEJahSaWhRaFQsGeQ8XFxXj66adx5ZVXIhaLYXBwEAM
DA/D7/QgGg+zrptFomC5Ak04y6k0kEjkD2EAgwHYE5KVGg1AATKEg/yIy/qRJnU6nQzabzVHieb1eVstZLBaUl5e
z/YdIJEIwGOSsy+Li4pw25HIfp4u5fy/pKr13717s2bMHTz/9NDo6Omat9O9+97ulvO2SYTAYcPnll+M//uM/
YsG/fPqhUKv691WrFjTfeiAcffBCf//zn+XWSiH/961/HN77xuhgSxci4UJ
VMrFYnGO+RscbtaucTicGBwfR1dWFoaEh9nIiHx4hRCJRTsQKVYVIZkrLQvvRfr8fg4ODC/pbmUyGpqYmNDc3o6G
hAVarFZlMBvF4nGevFGQpkUhmha6S4ZpGo2HpMF1EqDVCHCMK7/T7/Tyz12g0kMvlmJiYQHd3N6anp6eTwH73vf+7B+/Xpcdtllc37PiUQCgUBg1sVqZmYGx48fZ4fmlVSakYu2EOQqXciOIB/5kuuzOIt8LHbCT8T7pTp
gLxZ0fSKOGk0ozjSs1v5aLJY0MNLpdAvm8qwWvv3tb2Pz5s1Ip9N45plnsH79pTfmE19bWVmzdupX5SrFYDEePHsXx48dzLvz
CwV9RURGrXDKZDLv0Uq5gVVXVslXlgsEge7HQgJVAzsA0847FYigqKoJKpYLJZEJbW1tOO5eqkDKZbE611moh33G
e1o3iOYicvlD4/X6Mjo7meOjQIF4oGhCJRPD5fNzG8ng87CReyMSx0EIRKdT2JJ+h8vJymEymldhdZ/EuxJJaaWs
RRMC+8sor0d3dnaNK+8IXvgAAcybOj4yMoLa2dtbrq8Uxevzxx/GpT31q2d5vLYL4C0JCX37em9BBm3gURUVFrEq
iUrXwkNVqtbBYLKiurkZ7eztaWlr3eztaWlr4xkktLpqVxONxHhQRCTCdTiObzUKhUPgjy661LpZKObjGFE1p7e3F0NDQxg
ZGYHdbmeSJtnnE+GZJM3CXCkyiZuZmYHb7S6oBKLWXWNjIxvbjY6OYnx8nJWIFosFbW1tsFgsiMfjGB8fx4svvoi
9e/diaGhoySqQ0tJS3qeLARnckYLxVFBUVISqqiqePNAxA7xtFEc3bbphx2IxLunLZDL4/X5uMZzKetTW1jLLBlt5
fKpWipqYGFouFOTUqlQoajQbV1dVobW2FTqfLCRam/D1qcQAnjmVqWdLEKRgMwu/3Y2BgAMeOHQEqAMeOHYPL5cuqptMpkMuY
GAuBJDbVXCDQwViqVTNIVDmxLS0vhcrnQ19eHvr4+uFyupX9hy4jS0lK0tbWhra0NtbW1O1OHjwIEQiEZ/vFGchlUq
Zw5W/EKeLjAyTyWTOtYfOQeJ66fV6mEwmVgsKJ7WkcstkMpwhl28YTBM1MpWk9yaVrLDtRg7p1EoeGBjAli1bmFP
W3NzM2W/RaBTj4+OcEpDNZqHRaNhaNhNnawSyACS1ovc44kzRccFKThJTSaXyxEKhXJSFjwezywSuUajgdVqhV6vZzN
Nyo7MZrNsDSFsQcZiMVx33XX48Y9/fGZxzsNYiFhILQpiZmcH27duxZcsWO/O///u/cecq/yTMpF
IIJVK5cyi6ASjmZjX6y1owU6uqUQ0nqsiJVzyJax0oyaptHC9AeRI+qmqtFxcNBo0AchRbJzF8iCdTsPlciEYDHI
+n/A4kEql7JSbTqehVCqh1WpRVXFnjjEv6KKjpCsT8/9fj9GRkZmWR1oNBrU1dVBoVAUtE5IJpM8qCV1TDab5QH
IcpvClZSUwGg08oCE1onCLtPpNA9uaXu9Xu9pi4k4XSB3ZVJEFSJNA+DAXIogohueMA9QODkS/k6YF0iTBiJLLyZ
u4nSCYpeIWE7iiuXEUpSBJAIJhUJrIuvvggsuwJe+9KVlfc8VV6UBwG9+8xs8xs88cQTGB8fn3fn/XFHjx4cCl
veUogAva
2bdvgdDoLkTIeeughPPbY3vnNZ5aS1lpSwHZupMvBKnBTgdWYvvWGs5u4+Lh9Xrhcrm47VZeXr7
kQXQmk+G2XjweRzKZzJEoCycWwoWqhVQ9OXz4MD74wQ+wQ+QQpDNZllhRj47tMzMzDC5nawkwuEwpqenMTw
8jN7eXkQBrIkiCCMhGp4lhfX4/m5mZIpVKupApV0HK5nNukZ1RW2n/8x3/

gX/7lX/CpT30KTz75JP72b/8WQ0NDeOutt3DzzTcvaaVPFYFAADKZDPv378cjjzyCrVu3siLt5ptvxhtvvJGT+RI
IBE5K/FprWWlLASmx1gpWOkdspRGPx2Gz2ZhQTmRjIdn/TN/GhWC5tnG5XXHr6+tPKcuPyP4bNmw4pe1rbGxEY2P
jrNeNRiMaGhqW/L7LgWQyCY1Gg8bGxoLbaDQa14wVy0Kh1WqxdetWbN26FcDaUN5R/hn5+hAfb2ZmBul0GmKxGBU
VFbMCtxeCQtsXCoUQDodZ5aXX6086yaDBEanSjEbj/8fee4fJVdbt4/f0XndmZ2d7L9/9KlPTlvjkSlOD+whZCpoAt6d1wBmuzPTL1Ly2QBFK3SHlIkSQLx7QXl0+4/cC1xYlq3gsZu9v6JIbEP8ZeYGnW/rOb/3jpwtlfdiMGmoH89XW6wxjMvFwdxeypx5eiuPk3yQdjRbEX79LHNlLL1rCGSXZL5MVYXxiVc61j+nIAvk0J7rqJnI0ZZxYNsnpyXdXXqLFCRmW2ic0A7bkNJsbXjmR3Dai9XDtDaKd2WTbB6ikjpXUS8Ak7P5cNcxZUzTEKG+yOB0kzbCrgh0MyzHrfrR8jFpTeqi0Uhp2gyKHoeGwEqrFJ3VCMp8xbCZCS3wvNJUJRpuTI3
UBBQRUdBXNOjLq1/lFUHpvBRpEhBCgABBIKTvpu1me9/ZMr3s9PL7I7/P45nZ2ZrNbgK5r+tcM9tmz5k55znP8/n
cBSaTadZ5cECy4/wpkZX2yCOP4LHHHHsM111yDZ599Fj/96U9RXFyMX/7yl2xGOtP4wQ9+wNQmv/vd73D//fdj3rx
5OP/88xGJRHDmmWfipz9Kfbs2YP7778fv/jFLzA4OIhgMMj8dmYLsViMcUlcLhdrW3wR2kRU9ua2BbktwOkCRR0
QWbanpwf9/f1JyjSRSJQUesh9LhAIWFI91yMqGAwyV+2Ghga0a0tbWlLTVLpVJkZmayizkej8NoNMJsNmPu3LlYuHA
hysvLkZ+fD5PJNOufLbPtNTiu6xOWBvtx41xmrLMb+jStmSIDn7eZBFsm2kcfBIiY+gKGtyFOJZ5/pUtrm7Hqxwkw8ViKf+4Iyh3NIJ+6tfydMbs6iyjXvUyfBl9q
hysvLkZ+fD5PJNOufLbPtjmJTeHZPbDNbyCC8JbeSxKqXaB4ltiIKDYu69XZgwDC+LnKiIkZ7gGNpqNfQWCjHKBvKvVnUQbgKhxbbJ57mmMIFCbkkCGTKM5hpyGkAH4Na+vfTU/Iknpe
Ohx9+ePr2cAJwOBx49913sWnTJJBNHb8cWnTJtxwww3YtGnTiFiQRx99FM888wisRh+8YtfAAqKysBjC7Xn6lIkNHI1zKZLCn
+gutSSioPAmXScB2aSZVGMR1kyc99ToRNu90Op9MJh8PBSKFut5tI5sZERI5MDUSRCaTJRGpies0NDTEuUu5LqnEuq5uu9H2RSITc3FwaNXEjjk0nEnYW6bdQ35jnuE0DLqn
cjRt1MdrvCIVClg49MDAwY0ZnpMwTi8YZnpMwTi8VwOBysXdnd3Z30e4ODgzhy5Ai2bduW9H2RSITc3FwaNXEjjk0nEnYYjUaEQiH4/X4WNUA
tpdTPm9SF3HIzldzpUSQSoa+vD21tbUxu7bHVuCTeOrUKIlNHvt7bfwYwTcOLfalxFpUzEFrVBNaRGrsAkHS/S9D0Hq2xFWAcWvRN2Hd9tvd1F4VPqE9PmyOl3mvKfd1UUuLZGHnLTtI
RI61nZmZCIBAwd2visHV1dcHlckGpVMLr9ccCpqQk9PT2sdUTu7skZmnhTeZz6MhNfLE5/MxVRq
XX8R9Tq0qiUTCFFKzHYxLTuJyuRwSiYRN8cgeEqoI9y0PduYk9cHB9Hd3Y09e/awqwoXLwL5WI5bjKZjEwFoEoTEz6gnorOaAQ5ocgiDndfw6L/TfgeZDLc9Yhlna0aGh
paWH5gUQGJ8NMrVYLkUjE4oGGh4eZRGI4eezNxnHz9
wuVzYvXs3IwH3HA78NHoRCJ1KIRKI6dpprr2HRokU2O8CiCaJAdjhjPEmdDiAcBiA1c5wGkk53BwKHDHbIL8RFtHI/EqZbbLDQrC8
77+Pqq6e+e8arRZCNBPv74Y6xdf4yuxZOp3PMmfgeR9RZCUlkWlHZldRAzC3m6/Nz0a5U0QYUTi62sJMXD6RlmMk
m2g7QZIwiQbg2CKkl6mAwyAZf6pcDx0dowOjWdpWz6qpDPgOItNxlBbpwIiNx+vEoxLiQSCUwmEKJV
KlovX39/Pgqn5fD4zhzbyAWBwAZ2u9mAJxaQLSjlsfLz63WUzLLAedOUQw0YU9VQpIyh7ZTzCZMgQTg
cTpoUUKI73Yy5+6bRaJCZQmJRDkerBHFzkmzErD64BKVkAiEQpEt1e9PT0oKenB0NDQ+g2Q0oOhiAwNtTTRQl55Ek0M3U3o
ShOfxMFpECtdGINVjLd33qQrMtRXgLuzUajXMZmjUUaj0xlKuJCSW0x0ICyeilZQyu0ajjQZ2NnJzc6FSqzhZhbOU1
46RhpoQ0cW/TzeDwmrCEBDinfwuFwWjI895EWdTSGhkIhCIXCJD4XbfYmXVf/0zfZ6cqUkea
cTzj5+jvf+Q7y8vLwq1/9/9Cn/729/w4x//GSicDn8+HjjjzL/
G+vXrASBpYEYENyKPnAcaIKGQUOSZJ7M0VCPqSsjNzIKLVEbpbLRGfBiyxhYNFQxvQHAUHhBvrjzMcoE/
0GlnAYTgcxocffjqVq1ePMIk9k9hfXgAMbMbBfFdFRMGvA8AcktnZRMb5SEh
386bJAddRNzX8NxwOw2QoasSkBCUljJSgtLYVarWb1Wo5Dr97bWb5Y+FwdcCwWFj
AvVFQqKNKpUJvb28ylmUlkbg/QNaHm1prGtWr1UhyYVQNoQAPHGgm+FMbAQkm4F5q8nGNOjY4R8nN5fDS0a8aVKIjR8RXkOgTUEhpLq4NgABsSGXK0Fp13yEjCPJu
A1tZWWCwxONx8g1h1ZRWrTJodLqHAkWhj2a6bv2sJu+K+MtAOzDMR0Ci23BmVrdyJc2Wn2UnbVbNzGdtwyM9
4+eabp/KSx4WysjLB9hHDVDGdchLDVDzz2bMHH3744m0SQNNkA/gzNa2btsjRpu8vsOazYKAjgkqm/63v9fa6aSZ1nhWIIFDm3J
kfiqIRCKsTH8qEsxVitbISmVSqYyuHFj0vdjsRjsCoRAyMjI
gk8nA4/EY6JgXhMNhdHV1ypD5V6OZ8xnlJYGpx64GqWmyjmYU6sa8Frx/36ZXtuQdSxtXZYKF2LdkJ3EjhR9KQTTJ8NJ9yS2S1Wz
k2CpbrVaz1SgAlixdxdYlzqRhRjse4QQpdBsMq
oM6+vry9pf8LhPZUKhVcYbJ0gupZhYK0gupZUKhVbZdP/Ki4uLmRyCkYtR3o/ME8W4Q
wEENDQxgaGoJer0d2dKMRuNJ13KKRqbYEFYQmMRuNJ13KKRqMYHBzEp07emll455jLFYjGUmUmP3dPMTPU/3M262JanKLBYL2tr
aYLfb2YTaZDLBZDJBqzpyHHm
0u91ueL1ej5gtHGZjdjbTKZjE1i2VnQ
OOPma3hjCV7/6VXz1q1+dyktNC/R/R58eqZxOSbDc74ZLS
RwPr169HR0fGrMRsgczNZgKRSAQA9M2mw0Wi4XNIBXVaGsMimUwiEAgwdMQoamocHDwFmQnPR3v2QD9x9wJ8jMqNDYTWcu0Dz/6+oHZf2+Pq72H7WqJ8ejyvDrPeZjCxYFtTF9oqMgDNqbnS3iZCtpkZUBfm83bHXqV2Wk5nW5MbynHe6W
ZzUdSnTh3tnEIhGXdrTxfl5LXm5XLz8z+Xu1fOCQI+2lhMEZXFKRMQTp43TLeGrOlmtTBwlR0S5rakWeObyUm1C2UzVIFBGPLNhKGR8+uxWLbOrHf0W2qFPeH3WCXOdUWrQ
2POD2fIVCIZvEXXLJJRgYNYGEBzczMA4Gtf+xq9v4Pk+79nh2QZVOeL63DX6zPGB1f7n42NwKfnYKmeGwfjtbtE5gSjwP3aR/FDf6gFjjtWtrq3hB9UwCpSbufMpWRw
anE31ej3jbWRlZUGr1Sa/5S5S/N4PEZ+JNLjaI/pvhcIBJjSowpGT6kA7mlXSzlMj0IMyF7HX4M6emuTWrrgo0qFkS+pynxhPGR3SUfPpnonSYnAp4z
xJRIJa9PY7fa0Ttb4AnAnVmKnhnq9AUo+BX16sQ8xgVE80JP81KMyCZ9FIDTqCAOX7iwF6IoP+EApKgvA1Tv6t5nvXYTS07LNt3i1mKTHRxMQmljUKkjpJR2KXcb
FgkAgwMj4lAjeJ09PDApUpiHXRokVYtmzm7ylZu83FYrG0rMLAf+X6h6eTC+iVVH7DCGxaeJqFi4nIJ8wDXN0uLN5RLQkOqMGKBSotaudvZBDwhrvUNxpNP
ap3Eap3GiMKWJ0VtvvYWvf/3rGB4eZgoUl43Zgs3tUu6ytqamfvmO+OgLRLB/TGqjqp/3rnSN4e0qg1+xOqJV8Y0BCt/sweTmOvmCzYgdbuiThNUnaxo7M/f+7KvoIx4JsMnyOvNDvl
AoK6uLu3KhRXjFM3OiLRSFbjG3XcwjgXFN5eCmTkxNlyQEmrRM3nchwVkmv+bY2cyOWXxBpJ3KVmX+ESmFntgS0qbtU+69S2uBOpI7J9UxmL4nVOsXa2b4rLaHsxZz2HpoLPqFpmu8qTqS+vEjRkdrSmz7cVilRLQrRSq0oOMoEqFvNlc8jaL7D4rFkf4KfFPJQHfcKuLWSQW2D31oRuQwK8aNmLVaOhjfUz48HotlChM3N4sc+o4kLXYjLRjcz9LCCzLdl0e
Wi1FSUgI+n89e1PqIyhMDZW8o8dOPT8IuRwM2h8hqpsVOSXo6rJkfSBRjA1kqRz3FK7a4XWxZj3/P4v1TNZEpYHFjRAmu5RbNrUfG3IHkz6dM5NeZBSFQW2PO1zgzt9kCd6tJrXlq6t6fXAZBgBPYXvvkL3tUHFS0Qka2gG5wq2JKEYHMzbrY7oafmaBa2GxMbQr3MYl6kbS2bzjZbIvpAyvptQmw9TNnoKyvBlAdmShWxGPyFWCVSpCBaTqfeeUlnQ7xdjL2bzqkRWJQVYDXGlJ8jfl6aRUFMDlKhnA
BQRZqLBaLUVhYCLPZzFRWtJlMJuTk5LDFAXHLyPBRq9VOqLJttVpRX1/P/h934uzxeDA0FNITDhw/DYrEwU8tQKMT
S2zMyMpCVlQWJRMKqrUajkalcCZFIBK2trThy5Ag6Ojpe1mv10MsFie54pN1B6k46XkgEEAkElLaqbuROo
iLLXdRIaTpEojkjolAFBbmMuz5HL7Rntfab+5cUWppGTuRgudWCyWdiyZTKWYf/ej3GH73SeVTNxGEQiE

MDg5CIBBAq9VCKpWO4JOKRCLWIj9lVGnl5eW44IIL8MADD0yrOmuqmIwq7ZFHHsFPf/pTDA8Po7KyEm+//faohms
zpUqzWCyora1N6qVyTxKuvw59j3rR3PBW6iFTnk4gEBihKuDeLLhRDxP5Hpd8TZEP3B48PVepVEzSr9VqIRaLkxx
hRyNY09dcnyNuT5sGHm7vXiaTsXgBcvA9mduipBZxOp3w+XxJ9gQCgQCxWIxVnrjZQanfo0k7vR/03pE7LnFaRCI
Ri4mh80an0yEnJwcqlYrlxzU3NzP+w1QI0kajEfn5+ezmxz3vgGPVNKvVOmOWCqcqiEvCtQcgEKGd+zX3kXAi32O
pVAqNRsOy2k5WKBQKdn1MBFyLEALFME0XpFIpDAYDC2cGwFRdXLECd/KYCuILcSGTyVBQUIDi4mJoNBqIRCJEIhG
43e4kkQSp0sjigPLcaCPiPE1wUyv2tF/U0SBhBS0yh4eHWYeCO2klKwTKrJuI0nPNmjUs43S6cMJVaQqFAocPHz5
p+s2kSnvqqadQW1s7Kvn67LPPxr///e+kv9XpdLBYLGmdR0/1SJCTCV/04wO+GMdIqjlqTcZisSTfKrFYjH/+85+
ssmE0GpGTkwODwTDuaycSCQwNDaGrqytp83Aq7HY7UU0ZDFkBZtZYmBlt/dBdAsbFxOjCvWdvuz9EMGFDlKMuRJM
QcNUfadp1Uu9TjoKABiPgzAWfn1qlhiUUsaXvgLXAQYmdUI7wvGY6u5QYW1Uc9MAFrbZYMKodlsONedMn53uXv2+z
L9ZrPjQJKkqM2WORlUkp9kHVDfk2v81u0W9oVGHzNEtl36CwN6NqlXKd3d2HXYfb8O4oGTGnyDVyPNNo8eNKVG7V
fmoRXnMFx6ydi8M08eYFMQlTSa/s9U2bTvKn1cdBxaAoCgfEyHzf9jt8Ne4UD8aE4TYUHvzNHaZEr6fMbIPe6zxGo
ax6ZWTRBGzlc2Rx4BN9RPmdqn9BRkM6Y5eUvXfukI57lJ+2lVGFSnP7uJP7d8R2twa8jd6xqKhPp+C97vx7FXeSjeZ
Q0fR3h/Hi9FrN4J2EY9npenR8xgvIpr3r+qtkO/vv+eoQ/j9b7cG0m7UPGvYSIqlnW6ikvQl/btO1gNMfJbD6gEfE
zZ8jGQYnpMW0j7FBNZ6Q0L9d/Bgm2uAvFAzoeo4GmxvA/pjGWTPd1ZKebIVwA9wtrFH7aZuX4mt5WGFe1EPbUqp6fh
3FRk2qTtajN2R0VOWS8dpEkm3fgFzTSp8e2R+u0qi6m3bmaTHGbv1iqX9OgHhgtv+GWJfBjYkFF1AGTYA0oDnQtHx
Ioqn+ExNMovk5rx3ESoEWmJFFEi/p8IiPApEAGmyqtxCPb2ZPJFwjy5IiPMH9yB8yHV1vVJ3owsOBRlWN0P/BfcZQt
8ypy8Rgo7T4pQM3DUqZU3SWkrD3zGpP3wj3MzQu6xqnQ8tR/Jk+tBQG5Po6ZiOpcTEd5LyfAGw5j+o6DETM6U
K4TjjrmuI83TmFv1s/Qk9Xk5PEk7fX+fkOBTmVXYtAYFBqeCsobZHqZR9nFsMIG/mq6pDgOqjZpRmGW8o6TdWw+J
YZYNAqzU7Ue9HKwVWH9XqD+bBaDAZ8KLYZBtkEvNuGbu0HFfamT6JLY6qc0pVLZmSeSF8WqNyNwB/rDgFVqNL7X
+uxGGY6M4Ke9Evt+NhSvgTqGLcTYmN0NOMVuQPbGR0ngLXeN6Ywlj3bP5W6+QdfKWsEkW8uMA5v3w1MDtnGr+6Jc
TT2UxJjdVqGQZYbKFvCN5Tg85xvLWJiIEcXOtRvQP1qD/ER4mhqJkcFVZdDs2m7V7HkjOLSeKi2g6HwWUfx0u4Zh
5PQdpI7T8ReYSzI9Jmdq3lq8XxC/zvqmo5wpZnw9Fm9NLdIJyRKQdq8dEJVKw0Z4qQxUt3SfDKq7Qs2iXSj8hq7Lq
FLNkDhYVxAf3jsHbrl9p3qO5Y3fZ2sEQ5i+3YkZFw9vJWY7R+bVwRLGTqYhq4x2aIZp3Khm/b9RvLHAWqjAMGfXZ
-0wkGz9Cnw2lwbb5wLpB2p0UpSR4n7+gUz

hdDqZKoN7I6aqz+OPP57k60IurtT24bay2tvb0dTUNG7lQCQSsVBEOoaJgLK7qO2g0+ng9/sxODgIm83G2hf0XrW
2to5bwZHL5SgoKIDBYEjykIlEIswZmdv+Gm0j92GlUskmOceDeDwOm82Gvr4+SKVSZGVlsYlEf38/du3ahZ07d2L
nzp1oaGhgq24uyDSSJOV0vIWFhVi4cCEqKyvhcrlYmZyqjjSZ4baJeTweqxJRiKRGo0FZWRnKy8thNpthMBigVCo
ntMImXgrxTCbiqzQeyH3a4XCwFh5VMjUaDbKysk66FuRpnDj09/djx44dbKurq5twq1EikbCqu8/nG/Hu/HzhoYGvP7
666P+fUZGBpYsWYIlS5aguroaOTk5cDgco+Y2Hg/kcjkqKipQUVFxXK/j9XpzZJcrlckGv18NgMKCoqGhCHM/ZpIZ
MqmJUUlKCP/zhD6O6S7/++uv4yU9+gvb29mnbwcmgv78f1113HbZv3w6RSMTI1z/60Y9QUlKC6667DkeOHIHP54N
er8fAwAAOHTqEmpqatK83bdu24Y9//CMaGhqYx8xGmIdyDyDyNeBl0sz/VQBMGgUCAeDzOjPhG65fLZDLU1NR
g0aJFyM3NhUKhgMfjYSnUTU1NU3ofJBIJa6Pl5OQWM7vGxka0tLRMO/9EpVJh7ty5mD9/PtvKysqYY98mJgNfrhdV
qZZNA4hmQo+7777+PM844AxKJ5ITuByESiYxQCQ0ODmLPnj3YuXMna/HQvtIkqvqCggJlDEqeMiP8qlQoHDx7Enj1
72PngdDpZbAb5Bo1VZVOr1aiqqmKhwWazZGWVlSgvL0d5eTkbY9ra2uD3+0e415NBIbwMh4eH2XE6HA74/4k8j
ZJ1IFj+JHVq9ejQULFiR9Dh9//DEOHDiA8vJyc7uL3TFaLJtmEgkwqojxNfxer2ora0Fzp0
7097XuBFEtIChBWpxcTGKioqQnZ3NnLi7urpYhZrI58FgEIcOHcLevXvZmEabx+NQ0PDqPcEtVrNolAqKioYsZ+
qq6kxLqcSptsw94S10n74wx/i448/xt69e0c4c4RQcCASxduhRr167F//7v//05tz6cBzzzzzzDO64445xFR0k0X/nnXd
G/Z2ZIl9//vnnePDBB5OUkHo+PCJKljRQf3LbQRCzXCXK5HGazGUajeRKKJhK3ouuRuZk1mtVh8PtaW0mg
0jJxIpDuFQgGtVgudTjeir095WKcrlEIlE0Gq1rKJAG5/PR2Zm/PR2ZmJrvQ010YNKj4fD7G3yEF23icgng8zhb2
9cLlcGB4eRjweZ8Tm1OODejIwM5OXjWqZQO8TVRhSs53C4TRRNKvRQGo4bvwHmaaRQogIpRUVFSw
/arZApnDEeeNu3My6dF/P9kReKBSyBQKpYfOAmjONwmmieRMgKpYfD4fQ0ND4/5moaKouSCrvN
TSe4di8UwMDDAwpVtNtsIBIgLzOGPl6cHAQixYtgkAgrweDwcPXo
v72cKtYksxDIyMpIUaGq1Osm4USKRrigLzJOGPl6cHAQixYtgkAgwK233oqKigCOYPXoUf/3rXxGbWzu
rRmQTmRj19vaioKAAr7zysy8pL5YypmqmLU19eH9957DzabSXy8/NHvflOBNwbRcuWibRIAxt9PVMD3BchLmM
8TPcx+nw+dHV14dChCHB48CB7nEq7WqVSobi4GJZM+KysrUVtZZIqZQZTbIZQq1Qi8WCzz//HOvWrv8T//+PfffedzcL/vaCgY
xFp2dnWhoaEDbyUsL0PT5fIw8W8dCzSDBs6NZvdDhwmJvBm8RDj87G+6t8FUJkBhBXB9cnPWtn7qSN9Phtr+xRTfUhH62trUzN1d3djaamJi
GBgYQj8dhNpuxdGSzJ8/hGGmRyMm8Pdt9WLSGzu0cxlmkfncDiYnxFdP3w+H62trUzN1d3djaamJj
3txfAsYpCYWEhq1xx3J4ZfCYhUKBTs2vV7PZGkzhscHER7ezva2tbaa2tomdEOk7L3pBvezkMlkSbl4RPDPycmBTqd
Liv6gMYf7vaysLJSVlU1Z6UWO3IODg7BareeqxtEEp2axnySDCOqgDGOZKGPS+zaiMzkMqDMsFzkMh8MzGUGmN
gMIiPPvoIBw8eRENDA4RCIXJycgcs5hpaWlhESY5OTm45JJLcOmll+Kss+Kss84/wgjlFRURHa2trQ1NSE5uZmM
giwi/3z/h/TWZTExcsmDBApx55xVx3vvvlNUzqhPhSvRzlMgjPKBoEm1wFRcv82uzHsNn4RMZuSx+yl/K4OpqG+HAYf+xwD
AmzZtGjXjrLCwIRHQ05ODht8gGOS9wceeADvvfvfcevFz7sbXgl+uZsK7V1Vvfkl/hzKhc7yaWYOXSh6SpDXiMIMfHEgYOs
zzjn4wx/+gAULFkxoP0+Tr6eOL/rxATN3jH6/nxE+ThTbGHjq/W8sfLbK+1bY8YQYJyBlRg4Nz2tmuLzsYOi5LxLwMxgOmE7ky
VShEOh9lkYjoqNyKRiN34uZZOc1C31+9RGnQ4QB5GywriIRCL4+9/jvr6erz66vMNyYdlEoli9fjuLiYmRlZeG
MM85g7SmKz8jLy5uSIrKv145ZVXsG/fPtV1Y2wjFiyZAm+9rWv4a9+9av8cKR5dj+QNFg5JDyWomQgHFr9HUUg6P2t2TIMUUFe+vUB5YvZAVdrcd5ZqS2SNfF8d8bqHWGZ+RTCq8M3N5W6rs3TYW2S/dz4p6sG8pRcFrx3Xm5e++mLQdRnpTPjUEgb2rMR8MPZxaJGrQ2+QQaqNDPYiROsvPD
1dTVisRh6e3uTvH9aWlpw++wV4M4wXnmh6lyjNNJkg9SQHHJwImk4m1f+iRNrFYDL/fD4/Hw7yliOQ8ODg
4oX2SyWSoqKhAXVkZCgoKoKWDgcbdnZ2Vi4cOEIkNq4vE4Dh06N27d+PaPwPvPxcPj1YWz
duhXPPPPPMmOfPaNDpdFi/fj2rSm3YsIFVFVgwkTHWucTifzYDt69CgaGxvhdruxfpVFz1FZq6SMIcvwvJ2bUaslz
scqpWq6HX6zFv3jyV3kdfh3S8xfJsLxLxqGYscnjqVYrVlKIpjvfIMBv3J8e1dv3d3Czj52033VGrXGYgsrX0kPx0evV
nZydWrlwJrVaL//pv/j8Iwk3ZV8eZbB3UbpinoWcK4w8sXx8sphiZsmbYnnVsyZWXvhxbV8bUFhjQJGCsr8r+OVbJoVcLH
bXn31VfT09OZjQDu0o2NDUyRoDZHCEnbmQ1Tc1PDbSMVp3V2LmLEuzKJnBJSSDFLHwKMV3Lrkj8+nMW17E5f0pyxXYH7YBneNmY
bKCXeU6Xj6NGj43o2KRQKxGIxVjkm83Zi5UE5HmjYmRM4lxVUsHqlsdSZlicdCpqUG2rRT0y5RIyDKLx5pAOaVeGBh5zWw8VyZkf
fz2fEUYlEwgjr6VrWRqMRV155jJatWWt8mRR4TvKMfktepfsJmXnUNvVRfj4bfA+xTo8ByupS4LFTsw5yefWbshyZVaCSnXHAfDS
34YwzzpjysUciEbz7rrn85fj6DRN6rPw2rFnlmwn5tiBs3bsTXwOJ/G9/HH5/kuwFothcAvb29TK
futEEjdRP0yEPVygUSa7dRqMRmMr5Kmf1yGDnObwyTrJmCXPawCJ8Q8gHZp5wzn3mgNxoaGiAxmu6LMXd+5c8Hg
8dv2R0Wx5eTmysLw73//G5bnsvFP6veWj+XJA7D66xj3ZzGWhJ7YG2GzEcQWfU0zTXnPK88xA9RXvSQ72ZkC2BfKmKbBjq
xWXw+n010xWIxAoEA3G43G43+vv7rTpFAqFusEw1Wr13A5OamaRUJKzCwCu5u8eZjaaXcg+COOk8psVs3NWAC3EpS9wBxCh1
Y8fd+vx8FBQVYtWoVNm/ejBVzJGcJXzIKEgD0jRzCnfaIxsGmAdtJpK+3kHqJGrK7YBowXYZ8ZiaExqJQdrbs0jy9Zw47Dd0CgAq
UxMk6aZhOnOrHSF5YjY2NcLlcCIVCEAgEjFhzhKUQ+nyjEmEgns3bsXL774Imw2GC3Obbmlpwd69ef0f1U+HzxbjooCgqi
wwx/+EIsWLYJWqx1xU43Yr0xRw8qRe3VmeQsaNRdQtsG5ZvOYbttX3F0xZHZuk1Ye5dx3k+QoSXDX/VrQOcv8h20gC+CT0s
Fi1FeXj5tvEY6T88//JF577bURLdQ5eJ0Jg4etdAUHBPttvdmcxbQdmHAiiYnNmjuBR2zzozwwk8LlLuoTt0W7Su3C25+qg
xdV+Oxx6krq4OdXXV1zIdt9+7dePPQN7F3714cxBsLbu54A/6kUzps3DxoCp2BZ09PWhubkZKdzZyeAYYKlhsXqX4vrrr8r8f5558qU
nNBrFjh07sHv3brS0tgDXXrl04cuQI+/k111yD//QBYBhYSG2bt16Q2oWnjZrjuJHqBhCuWKwE7b0QMMZJguWwQw7Quw7UHZZbcW3
tmLEpIpGIcZ1o0hQIBBhHliZP3+P/8nR0zdOL7H3Bl2Rv4PGTLZs6pb3IJiLpI27OT3/607Q/p9eoqKiAwWWD
AAw88gDPPPPBOFhYW45557UF1djYKAgrR/O1ORILFYDPv27cO7775Jd5PjNmWL0Pv/fXHPwEDmNtYsWBDbDA0kg/Du2NqezFlz
2nq5sphONkzFKYrrxRThGvV6PFStWoKqqKrBqvdrk45psKHDEmalgeLSTNzwKHXdbPnPOx6HOuPYY1KnwXmSuMEXSqa4ndNMdlKHLbMkRAwDBK3zRvLQ5TEQjePzgHQv9MFjGM
BFVdcwQbjWCyWlqtTVVWV5On2l7/8Bbba2vb4kcgkCwxkRW2D2u2MRsAdVVb0sl46d9bG5vbnZS0Nd9swGWD9Nv4cW6F4uu+w/PK
Xv8ScOXOS/t90RyPQwvChhx7CU089NUUuxqlN58QnpCFgeYyRE3uHvKyFObdk6Dcas0iVnPWj4IND1euBkFOsK4Ln7Ld6Pp3xwI1O

BWq3G3Llzp/z3wMj3ZrTz1OPxoKWlBXw+H0KhkFUCP//8c7z44ovYsmXLqBUsiUSCW2+9FXfffXfae1MgEGB2DgQ
Sz0wGK1asSLoGDx06hEceeQRPP/00/vGPf+Af//gHq8B+73vfg9frRVFREaqrq5GVlcWcupVKJTIzM7F+/fopEav
5fD7mzJkz4jwcD2TXQiR3mUwGlUqF3NxcZGZmjlm5i0ajbGJ1ykSCTDf27NmDZcuW4fXXXx/VAgAACgsL0d/fnzR
rfOCBB3DbbbfhwQcfxN133w2HwzFuO6++vh6XXnopOjo6AADl5eV47733kJ+fn/b3Z0qV9sYbb+CZZ56Z0t/K5XK
UlZWhoqKCSYVPxrTi0zgGq9WKpqYmNDY2oqenh0m2KW2bWgIFBQVYsWLFtPjxfNFx+PBh/M///E9ajymxWMwWQj6
fD7FYDFlZWcjJyUFVVxiRwmC5LoC4VCRlzv6elBS0sLOjs7UVhYiIsvvhhms3lCr5dIBCNRpPGxUgktraWvz
73/9GX1/fCGUknWt6vR4LFixATU0NTCbTiKqG1+tFR0cHent7WUVsYGCAeXOlthELpNnmZmJuXPnYsGCBS
FkOzAwACOHDnCSPE9PT3jigBkMhnKy8tZH1xubi4WLVqEpUuXQq/Xz9Cej0j0RbWxueffZZ1u2YKMRiMZYtW4bVq1d
j4cKFM3rOn0w4Yaq06QZlnD311FQZlnD311FOora0dlXwtFApHrNjMZjMsFgv++/w8ccf4+GHH8Ynn3wCt9uN/Pz/Pz
8JPJ1IBDA2WefDaFQyKSZfD4fCoUCvb29aVtWM6VK27dvHzZu3IjKykpkcpcddVVWLZsGfLy8iCTydDV1YW2tjZ0dHS
gvb2dbf39/aMqLZYuXYXYqrr74al156KXJzc8Hj8eDxeB//34mbaetu7sbO3bsQEtLC3NfzfcrKglQQqRV9fHw4ePIj
h4WGsW7cOF1544aajVtfHwRVOlJRIJtLa2MjKxSqVihqLr1q2D0+lEb28vU2zV19djz549o5Ic04HH42HNmjXuHNmjXYuHE
j1q5dO8KrZjZwMn2OsVgMf/nLX3DPPfcgFouhqKgIc+fORXZ2NlssnnHk+m4xsN9fX1eO211/Daa6+hqam
JmU1SZtZklWhCoRCFhYUoaKShAdT0SC7xQCTQ0NODtt9/9bzCTQC3NODtt9/Gli1bsHv37hGTda4VvGjZhHIrX/kKrX
SCXz00Ue47777sG/fvhE/N5lMEAqFcKmbDeW58p0T5yNTh+9jxB1f3tTyN7+b+FSLLs4MMT55CwAFYayzGY6DRpnrf/G
4ltN58FTs+4ov60MbPFCG/xfedwl/01mHMQ24l+xhVjWj2ZQjsjvswnfF4FgfMyqj1/hP9m25Mkjhm9K8llP/+m
hcDiYu21eXt6o5GuhUIjpz/yGiwcrHjVs0EdsWeRf0ynyvVbd9KbwK1xBvVn4/3UBL5+sknn8Qdd9wBlUq
FP/zhD1i2bBm8Xi+WL1+OZ555ByWYkmVX7p54vhmLl4bfqGYDjB3NHg8zWzuadm0/zMRiwd8Gp6pg6+xz+9i+yn/+X/J
TyYSsrCwcPnx4WshY2ZyPuUodCCLBgYlBUgkw/QigqvJTcnPxoGcxlgm97QTrcMGd9wkAcf5ZbdBSn1c/KZ5Egp8exBvP7lK
mtwkQOz6sgE3CLgV7Ua9fYBnE0HH+wiGMcCqPXgYLFCx0sH6Ca5OtEwH48IxnsEg2OIEEeB9DlBVE3+lVvLPUPYMfLsS2YqR
rmS8frt0mVx/0h9eKXHOQFtxRVBgn+W6KSYl9WDU6NiZ1kIWMCoWAxmDtMq8U70Aaw+Y4YHzcyuWUZtkxqS+aEPYhuF
BKT4VQUSeRNAKpZvbrLiD2iN8s99JMGqiUYI53tl2EDRPqzqXkaDhNkxTzijZ1jMdTRiVOIx2VZZ4Nthxm8i7G6kmg00
XDNpOJitmxKkxDGlr8f8kVmUWvDSC7LVMMVWi3zd2z1GCWBfOJ7x5erIAqjcqmy2bfaaMa/4jMngIDnuT5MiXhdfrSfDRBU
2Ebm/d9aqLTu6enjH0EaafGt2b0yfGxtY5aKzA4yBHKnVKEcPSYgDxnH5wpFkLp0BajbKfHRkJkxaAs2G9jw6ChNmNuPCN
qKsFdTwjsXf6Gu9M/zrk8NjvtH6frFnkbFaEJhAnqhHXtkxOwhVSJA8TpoNCHhF/mCPZTeRsPFwL/23Ldo97cRe1f/u
6rqWS0vHTQpfL8KI/eSYuKhpS3uSy/4gqNWJPQf2T7Un8/vVSutsY6GHzmcEoNPXbTQw9rV3HHOvSSpIQ2wQOzmJfgk/Wm
o9PjVoPBpDeJEP8zeDXTnG0G82n2p1QPLDBWBbf04cD7S2cAlu8f9/TR/XK2v/bZNj3DCLyGQdOgBUWoyXAjUw6s7P9
yxBKhufWFWF8/q1fzwe8WakwMuFHW8Evxvu1gJfDLAIF4x7xrEHZmXWcZEA1cJEdwjrMi9iXCAkU9WD6gaqTmvKu9uzsEJ6u
U89kH4+Aqt6/4Nd5OuHZHtHPxxt5G7hzOmD0Y7ZoymV/K3vjPSYmTZ41HSC5ajD61ynsaK+wP/3o1/f14R1XKGhaNgEs+
gI+lSZm58ncNwuBa1HhgzPr6CAXHwhwQaxcdEY8L2Nyky9WaWkwrfO/Pc/lYntnN3Jxnu6GZIh3NZl9xSTVmXavrXTFt
2ybzhvZpWcYtz3cx/yNUyEdN2IpgjLXsCeeoZxxeJ1PGzAPmdqoR2QPQR4/1CBRzAZxEnFPhnKIxRR2A2W6Xs8xSNBx7Yz
UVdsvQr+OdIyW3y9thR3HmG3Hc1wfG9W0iZlZy/HHJhutL0h6Rrz10E27Ja6d3FJcXKQMcAmqIEY2KEDqy1VlexBRN2xiyQ0f
Ypx8GwLjWW2PJrbXO4U/mS2NGP0SoXqwqu4Nl6sbgGxudh2wpAQR9Imgwy3jVBaP4I7Gh/Fb72sp+fqs6FRhKGzRX8QQNnqzq
hpZ3A2JeUTq9Fz/lmcd+JXsWxK5E/ez/kZNP/OVPfXeZzDh5vKm47Kh7sVlTmZ1IKCt3qBq7P7cy5LvKq+tuAm7E/vDCK3y
Dn6mdMc59KeNhnH5ChhgwQ6duFaRRCbFLGtNtnO+hWwQ1pwDCZLYKfhsWR5zdk54NWMD6z6yzLl6Bc6dVQ8IZvyhDc8hEm
RRuAjvxsCfMmaMKjMi

kkktw4YUXoqioCAaDAVarFT09Pejt7WX+V8RvW7Ro0Qn5/ICpfYZutxvNzc2w2WzweDzwer0si+31119PqrrxeDy
sX78ejzzyyHEvqKeKLy3HiMjXmzdvxle+8pVRf6+wsBDXXXcdbrjhBvY9g8EArVbLWmTjqdIeeOAB/OIXv8B7772
H9evXAzjWhsvKysI777yDDRs2jPibmVKl2e12tLW1MeUIbdFoFA6HA1arla12XC4XvF4vQqHQmOoKHo+H/Px8GAw
GSKVSDA0NYWBgYFI5ahMBVWmIyC6TyRAOh9lgF4/HIRKJUFRUxFaCZrMZtbW12LNnD1MJabVadtwajQZarRZVVVU
TanlZrVbYbLakihGfz4fBYIBOpxtRSevp6cFnn32GnTt3oru7e8zXFolEyM3NRX5+PmpqarBmzZpJX6RUsqZMNIv
FgpaWFnR1dbEq22iDN03eFQpF2kelUgm5XA6fzweLxQKPxwOr1Yr29vakmwxNLLnfUyqVqKmpwdKlS7F48eIpTfY
TiQQLNvV4PKisrBy3UhYMBnHgwAG8//77qK2tTfs7fD4f2dnZyMjIAI/Hg91uZxYWqZBKpVi2bBm7EUmlUhakeTL
y2WKxGIaHh8Hn8xGNRpmDs1qtRkFBATQazUlHhJ0KBgcHk+Js0qkGZwISiQQGg4FJ87VaLTsvnE4n492o1WrodDr
k5OQgMzOTOX2rVCoYjUZGQ/giIhQKYe/evXjvvfdw+PDhpJ/NmTMHF1xwAfLy8pCRkXHKZeURThlVGpGvf/azn8H
r9U5KlUaRINRKe/rpp7Fly5ZRVWlPP/00brzxRvT09EAmk2H+/Pno6+uD0WjEpk2bkly1CTNVMQImPztOJBKs9UT
y2fr6euzfvx/vvPPOmFUU4Ji/ExFIDQYDNBoNJBIJRCIR28hQTaFQwOVyob+/HwcOHEBdXV3acvZEoVKpUFNTg9L
SUkgkEni9XlbWzsjIYCs/4j5EIhFmtkabTqeDRqNBaWkp5syZM+bNmCYmQqFwxO85HA7s2bMHAwMD8Hg8rNSdkZG
BioqKERUbADh69Cj+9re/4d1330UkEmEE1crKSrjdbhgMBtjtdrS3t6O7uxtOp3Nc3gCfz0deXh6MRiMUCgU8Hg+
OHj16XBEM+fn5KCwsRF1d3YQmxJmZmfj2t7+N++67L+kGYLfb8eqrr2LLli04fPgw+7yowph6HMXFxbjyyitx551
3QqPRIB6P4+DBg3jrrbfwwQcfBlcDoFAAL/fj6GhoVEJwVKplHG7VqxYgauvvhoXX3zxtJkNngq
qtOPFbBxjPB5HW1sb43y5XC42UVIoFBAKhfD7/XC73ex3qFUZj8dZxiM375EWYgqFAjweD6FQCNFolHGdhEIhvvvv
d76KsrOwLMdHk4kR+hkNDQ/j5z3+OF198cdQKHgUNK5XKpIljdXU1E5ccb/HgS1sxAoCzzz4bn376KcrLy/Gb3/w
mSZX26KOPoqWlZUxV2vDwMLKzs+H1enHRRReNUKW1tbVh8+bNaG5uRkVFBT744AP85S9/QtGcxtatW8Hn85nscTy
cCuRrQl9fH+MXeTweFBYWorS0FFlZWdDpdMe98olGo/B4PKwlEI1GWVioVCqFUqlkF4/VasXOnTvx3HPPsbbJdEI
kEiEzM5ORs2kjYz4q0QsEApSXl2PJkiX46le/io46PkSjUYRCIXi9XtTV1eGjjz7CRx99hPGxkMbGxkkFM3IhFov
ZinTJkiVYtGgRSktLUVRUxMJduYjFYujr62PxC9zH1O9RW6KiogLnnnsuzjnnHBQXF7OMq8bGRuzatQt79uxBT08
PLBYLenp60k5wKW9LqVSira1tREtkssc80QDP1H2oqqpCTU0N5s6dyx7JfuJE4WRQ3Z1onD7GUx8zcXx2ux1XXXU
Vtm/fPum/1ev1uPXWW3HrrbeOyuFMRTweT6pEzSbH6KSqCxKHhPg/3DdJKpWm5czweLwRLSUejweBQMAyfIBjhL5
LL70U3/jGN2AymXDrrbdi69atC8vx9q1a0/sgc0CcnJyjsuNlpBIJOBwODAwMACHw8FkoTKZDCKRCBKJhE08SCl
mt9vR09PDWgQ0kcjJycF5550Hu93O2l8A2IqPvDdoQqNSqVgqeSwWY+ZylL1D4Yd+v39C6qJYLMbUSM899xyAY7w
HtVoNPp8Pn8+HYDCIaDR6XEnwoyEcDjPZt0gkgsvlwmeffYZ3mHhcGGw2GWNUbhniQHp02lUqGkpASZmZnIzMy
EwWBgE7ze3l60trbizTffRFtbGzwD0pKSlBeXo6CggLMnz8fTqcTnZ2dCAQCkMvliMfjcDgcePXVV5l8Ox0PjCq
5iUQCpaWlyMjIgMlkQn5+PubMmQOFQoFnn30W/lQqFQusnDt3LkpLS7+wbYv
TOI1TARkZGYy7u2nTpjGjPoBjrXmdTgev1wuHw4H7778fDz76Fb28
v8vLyUFZWhg0bNuD222+5Fb28
HtVoNPp8Pn8+HYDCIaDR6XEnwoyEcDjPZt0gkgsvlwmeffYZ3mHhcGGw2GWNUbhniQHp02lUqGkpASZmZnIzMy
5iUQCpaWlyMjIgMlkQn5+PubMmQOFQoFnn30W/lQqFQusnDt3LkpLS7+wbYv
czz333JezlTaTkSChUAhLly7FXXfdheuuuw4ff/wx4Q9/mLbXOxEgoi8N9NzeM1X
46GahUqnYYKjVakfcTFKfp/s6Ho9DKpWyalIwGEQwGGQDI00YpVIpJBIJ1Go18/ShdG0KvExFLBaD1+vFgQMH0Nz
cjI6ODoTDYTY4ExdJr9cz5QuZ2U0ViUQCXq+X3ZQsFgscDgc7Bqq2kb8I3fSdTidrEdINjrgYtHEnepR4TzcMuVw
Oi8XCOFTDw8NQKBQwGo2QSqUIhUIYGhoal4R+oiAWixlhn0zpvszQ6XQQi8VwvOp0jqm0kwybStt/vh9/vn7TT9Rc
dNNn0+XwTOp9EIhHjNdJklfsYDAYxPDw8Kt9Gq9Uyo9bUCJaxnicSCXYNJxKJpP+ZSCQQDocRjUaZ0AL4z9hls9m
YepU8snJzc9l4S2MILYiDwSA7Tu7YSRvxtQKBADweDzOjJEEMGQp7vV7w+fwkH6RQKMTGK+oE0ITabDYz2wfiWdJ
CiOtNl4q8vDz85S9/mdLnPxomQ76e1Vo1nbQT4QzcddddI1Rp3NcYD/fccw+qqqw3XXXTXj/7rnnHvz4xz9mX1P
FaP369dNaMVq+fDkuu+wy1NbWsgoZnXCUPEyTAno+2vf8fj9T8DidTgQCAUiEcb3IWIxbbSCV6vV7FEgELCVY1Z
WFjIzM0eQlilzicfjTWg2fzKSWidi6jkZzNYx0kqfnG6nwsEhHlh9fT2OHj2K3t5eWCwDA8PM4JrdnY2jEYj2tv
b2YqZVu5z585FWVkZQqEQ45pRu5Oe0yN9f2BggIkHCLm5uViwYAFTDpErOBGuY7EYuru70dvbC6VSCaPRCKVST6
fD4/Hg/D5XIhEomwAALBx0cHIRcLmdWCHPnzkV5eTmbuA0PDydN3Okc8Pv90Ol0bMFBCwFy5iZvIY1Gwxa5oVC
IqVxjsRibgNhsNvT397N9ov3r6+tjizfgWHsnNzeXiUjoMxoaGsLQ0BDcbvekQsH5fD473yKRCHuvxvtcThQCgQB
cLhcsFsuoCs/ZRH9//4jvTdTA8rzzzpv2itFEMasTI1IL7NmzBx9++OGrqrTe3l5s2rQ2rQJmzZtYn9LqrTy8nIAwJt

vvjmmKm3Lli1oaWnByy+/nLQPBoMBv/jFL9JWhqiMmwoaqKcLJpMJer0eVqsVFRUVJ83EYTyMVtofC9P93p2MmOl
jnI7/lZ2djSuuuAJXXHHFmL83nYTISCTCjPNkMhm0Wu2EiZrHg2AwCLfbzSY0KpWKTSZnk7TrdDrR0tKCSCQCs9n
Mqp5kV0CmldS2IxdjbuU0ndFrKr7IxGSqTnR3d6O+vh5XX331mIrVQCDAvNq47VHuo1KpZNVv8jojEF/P5/MlRa+
M9pz7vUQiwSrc3Ip7LBYDj8djk0CXy8VEEiKRCFqtFmazGYcOHcLSpUvh8/lYMLXT6WTTmu9Sypv2myo7P50t69Hq
9bOGiVquRkZEBiUSS1GJNJBIsAocqRQKBgCnTuCo1Wlzs3r0b77//PvN34oLP5608vBznnHMOVq9ejerqamg0Ggw
NDWH79u2YP38+OzenazydzGvM6sRIr9dj9erV+O///m+Ul5fjwQcfTFKl3XzzzzaxaxHuXHjxhGqNACsenPjjTfioos
uwgsvvJCkSrv77ruxefNmf0tb38Lhw4exbt06ZGVlYfPmzXj88cdx++2345ZbbpmV4z+NkxuJRAIulwt79uzB559
/joaGBuZKC4BxDPLz86HT6SCTydDc3IynnnoK4XAfrSvKZpv7y/7kwOdPsq2DZHoWc3vJKbaZvYUCJBB+3dr+XY4a
e1WrpthnUhDYp/Ub2ETmGnOs4HjL6ik25pn86etbw5vFJpEzyJZ/ ...

tbW1obm4e4bxLz6lFlJmZiVgshmAwyPgTRLieLgSDQcYH41YUyJGY2+bjblRd44KciLmciy87qBU5Hkhwchpjg8f
jsepuOpSUlMyoPQaXUnGqYVYnRq2trQCAP/3pT6irqxuhSuMO1G+99VZSKZvI1+Rz9N577+Gvf/3rCFXab3/7WwD
Aq6++CqvVihdeeIG9htlsRkFBATo7O9Pu30yp0hobG/HUU09N2+txQZOs4yXvUgYUOb3SoMYd2FInUBqNhhGh9Xo
9AoEA8vPz2USPJm00uSMHZ9pkMhlUKtWIVTJ3RU0rnIKCApSXl486KBwPaH/Gw2gqCrFYjOXLl2P58uXTul+JRCK
JnB0IBJCRkQGtVpu26kC/n+qO7ff7sX37dmzfvh2NjY1oaWlJCv8lkALOYrGMcPtVKBQoKSnB8PAwaxXE43EW80F
VtQ8++GDEftH+Dg8Pj6v4Ggt8Ph8FBQUoKipiaisytSNSrUwmg8FggFarZStaeu9owtnT04Pvfve7GBoamvK+5OX
loby8nCnniI8nFotRVVWFsrIyyGQy1s6VSCTIzs5GeXk5lEolU6VRi8FoNCIvLw86nY7xOyjbjbtQisfj6O/vR3t
7O1OIUWuYmwtnt9vx2GOPsbw/v9/P2szcqkE4HAafz0d2djZyc3ORl5eH7OxsDA0NMZVlfX09urq6MDw8zI6ReFf
U6pVKpRAIBHA4HEyhZ7fbJ7yAkkqljIRMTuzcr81mMxYuXIgFCxagurqand/0+VN0CvEWT/XJ6amsSnM6naivr2c
ctmg0Cq/Xy3hKXq8XlZWVuOCCCwB8CVVpRL7evHkzvvKVr4z6e4WFhbjuuutGRIJotVpGvp6MKu2//uu/8MYbb+D
AgQPj/t5MqNK6u7uxa9eutKRq7oSBJiGkriKlFU0wKLWYwmHTWa9LpdKkYEci33FjC2iFJpVKodfrmdU/gcqw3DB
Fkq9Ttg/JbE/j1AUtALitTvo+RS8QJ8JkMqVd1dOgZ7VaWZQHDYJdXV3MLJMgFouRk5MDsVicdC3QjZ+ECjTRpxy
nicYMTBYSiQRGoxFCoZBN9uj64C4OuByf6eCkncbUQdYl1MZJV5XS6XRJuYpESOaqV+k5jcUCgYBxEmkyS5tUKk3
KWuRGrEwFkUiExTnR64fDYRYfQzYo5MqdnZ2ddkHEDTnmqr/GAolfuFlm0WiUtdzIMiEQCDBiPV3TNLkhkZBYLIZ
arYZYLE4KEB4Pq1evTorjmg6cMqo0Ukb87Gc/g9frnZQqjSJBNm/ejMsvvxxPP/30mJEgdrsdX//613Ho0CGWlv3
d734XDzzwwKhv0kyp0oDpz9mKRqPsBhKJRJj6ZrYcf0/GrLTpxuljnDyoggAcm4Tk5eVN6GYSCoXYhA04dr5brVa
0tbWhq6uLZQiKxWLI5XKmbCSVnNvthkgkYhUNqm6IRCK0tbXhoosuQnFxMWtvTgZut5up0nQ6HUwmE4xGI4xGI2w
2Gw4ePIiurq4ku4hgMIjOzk40NzczPoZWq2XVvcHBQfT19cHlcrHKrE6nYxYP3Ju50WhEcXExDAZD0k1RLBYzB+2
GhgYUFhYiFouxGIdEIsHeR5r4SiQSRmymbWBgABqNhhmVzpkzB2VlZczagluJ41bkotEo9Ho9U+gZjcYRC8zU2xH
xEanqEwgEksJRqdrV0dGBuro61NXVzVpuWTrw+XzWcqWqn16vZ0piej8TiQRTAI7FqxwLlIMplUoRDAYxNDQ0Ih+
MJvXRaBR8Pp9V3Ug5RlzSE438/HxoNBq2H7RAp/vU8uXLcf311385VWkzGQnC5/Nx6aWX4re//S2ef/55vP322/j
ggw/gcDjw4osvpt2/mVKlnYjXFolESVWekwWnalYatYMOHz7MpO+UVE++PHK5HHV1dYx/QtLo3NzcLxxHYro+R5P
JBJPJNKX/n/p1fn4+8vPzj2t/iCe2ZMmSKR+fwWAYVXmm0+lQVlZ2PLsI4Phc6I+XmByNRpm0/2RDIpFAR0cHDh0
6hMbGRlx11VXIyspiVg1Ucezu7mbcOso6S63Y00ZV+VgslsT9S7elOuWTl1AoFILX60V7e/ukjof+L4HGlYKCAoR
CIYjFYgwODqKjo4P5NY0F7oQrFouxoN904JoDC4VCNrknrh0JK7ihwBRMTuIUv98Pq9WKQCDAxsPKysoJ8Y5oX79
0qrRUnMhIEJ1Oh/POOw8+n48ZfV1yySV49tlnmd/JbCESicDtdjNytEAgYBekUqlMMuai8iaVOOk5j8c77mOgtgf
5VZDiyuVyMZMwyrvRaDSQy+XMb4R8iLjmZG63mykSBAIBi+ugwFC9Xs9eg+uZQYOuz+dj7r+p8noqC3u9Xlaetdl
szNGbiOQOh4PxNEZTUVBadmpbhMi9zc3NOHz48KR8OLhQKpWoqalBUVERG1zI6C3VaJLk3rSqNhgMLEzyNE6DMJu
TkpP5XCTRRF5eHng83gh7EIlEwlpoS5Ysmfb/TxUu7jVN/K6BgQF0dXXB7XaztlxOTg6ys7MhEAjYfYfiqJkRHCIf
D8Pl8jKvF4/FGTG4DgQBaW1tZl0AikcBkMsFgMEAmk0EkEsHv9zNhDHVhUkUemZmZKCgogEajmfb35lTCrKvSZjI
S5Dvf+Q7+/e9/s6//53/+BwBgsVjSBgnOFPn6pZdewvXXXz/qz7mTorGg1+vZqrmwsBAGg4G4G5pVLvlwwBqR9Mzz0
ez4y6K5+K4PP5KCsrQ2lpKXQ6HRQKBRtsyGvE7XbDYDAwU77+/n74fD7s3LkTQgEcPHFF+OVV14ZdSCdzUgQIv3
NBoh7oVKpmAqGqlgulwtOp5OZoFE1h5thRhu9Bkng6aZDnAHqpXMrX1yyrUQiYfwQ7gDBfU7KtXQbVYPEYjEjy1I
1izgoFABK+XDc6hVtJpOJOUlPtioXjUZhsVjQ1dUFu93OCIxut5tlyJFyUCAQIBgMJrlZH49Ka6og0r1CoQAAtr/
cRQKpjCh/bCIQCoXsZsmtnCkUCkbmpJssxSF4PB44HA4MDw8jFoshkUhAlpdDKpXC5XKNeH94PB4TZhDpM110BnA
s64vrHh4KhdhqfDoiNSjja6Ig8z6q3E72f9H782WFTCZDfn4+c+AOBALs3KXrjc5VyqfTarXQaDTss6I4Hm4kz2i
PJEwgrhu5cnNfgws6NzUaDROvTCcfijhGNJGkayg1Y5K+pkeuOzyNkQDYMaXax9D3AoEAuzbpfVUqlWyhxl24icV
iZmTrdDrhcrnA4/EYr49rwAscc1Qfb6I3WZwy5OvZUKXRCrOpqQk///nPsWbNGNGjzyyCNpf3emyNexWAyBQADbt2/
H+vXrmTkeRQyQaR93hZSqoIjFYujr60NXVxfrn7tcLtZmo4oCt7qQ7gROJ4+fDpwmJk8NwWBwhH8Rt0pF1T6Px4P
h4WFoNBq2qs/OzoZUKmX+SRSFkZGRgeLiYiiVSjidTnR3d2PPPnj/2tvbx7QmGO1mT0GePp+PtS+5E1i/339
Cbtp8Pp9VwoaHh0flS0wWEokERUVFyM7OhsFgYNiUSCOY3TZ0I3CprU9vT0oL29nd0sKKYlNzcXsVgM3d3dsFq
tSXlY6SCTyZCTkwOj0Qi3281iVUj1Rm3oVFNPnU6HxYsXo6SkhLWlaaMKYVNTE/Ly8ljFRiKRsJs1NxaCbpZUhez
t7WX7K5VKWZg0RX3o9Xr4fD4mybdarUkLIKooktcTvY+0UCGlo1AohN/vh9vtQ2y5GRBOsoHQaF
QsIlYKrjqNnoNpVIJs9nMPjuHw4G2trZ2xGfPHnz44yejqtJ6e3uxadMmxadMmbNq0iyyCNpf3emyNexWAyBQADbt2/
cU5UGpO/LNzQ04L777oPZbB7v3xs5mMBKGVR6oBnEQimbANgclkwqJFi6Ztv04ETlXy9WQwnccoEomOOwNuMiuvcDi
Mjo4OlgmXSCTYjS8zMxNisRivv/46ampqqIBAIoFKpWFDvWCCJP3fiTpvP52PO2MXFxSguLoZSqYTT6UR3dzf
YaSJIlTyu8y/FndhsNthsNjZBS5XUJxIJppCijXx7duzYgeuuu+64+64WnLBYBD9/f2sIjYWiMvmdDphtVpZu200RRx
VV6lF4/f7YbPZmEmpSqUa8Xe0P/39/eju7oZWq1VV1yBwsLCSfGVqB1HOVsnCwE7EomgubmZOObD39PRgcHAQEQK1a

sgNlshslkgkajYdVrq9WKwcFBtkWjUSaLp3GfG8XDVS7So0QiYRVwqhIBGNEilclkEAgEsFqtaGpqYjl/arUahYW
F0Ol0rGIai8XYRHssgnsqx2h4eBiHDx+GzWaD1+tlEyiJRDIi/DY1FDcQCLBFFlWPiKg+2hYKhSCXy2E2m5GRkcE
y/cgyhq4/8quKx+PQarUsAoXutx6PJ8kaADh2ftfU1Hy5I0FmSpVGePrpp7Fx40YAwK5du3DZZZeNUBKcxml8WSE
Wi1FRUTFq7EMkEoFMJkN5efmkBhqRSITCwsK0XL7pAo/Hm7LKjUA3WK5lAFV0qT1CEyvKs6LqLlV6gWPVlKKiogn
9T4FAwF4rNWw4EonA4XBArVZDJpMxiTdNLt1uN4qLi5m5Zn9/Pz755BO888476OjoYHL/dC2b++/H1KpNKlyrFa
rkZWVleSgT1tWVhaEQmFSdNPJApFIhOrqalRXVwM4fuXdiQAtLtKB2kpKB2kpThUKhmHYD2elCPB5nfkyTwZc2EiQVJ1K
V9s477wAAK802NDTg3nvvxcqVK0/oYD0RvPXFXXna7HT6fD3w+H2KxGHl5ecwjhTxUqARO7csTo0jUYDi8UCs9mMefPmzZqf0vFiYA
A+/btw759+1jAMUlZzWYzWDuWVJpPebm5qKQ4uBjZ2dmsHTHW+0g38NlUXJ5oBAI
BWK1WuFwu8Pl85o9N4EgcDrS0tKC/v585U585U/N4POzfvx+bN29Gc3MzenPsorK5Gbm4u
uri60tLRgYGAAVqsVCoUCc+bMQWFhITMDTOVwWCwWHDhwAI2NjbBarWx8k8lkSW2liYBrNknu2iaTiRG76RqdiMs
3n8+HTqeDXq+HRCJJS4YWCATIy8tDaWkpU6+mblRpisVirMrFnVSerKA0eaqEkBcWVZqkUil8Ph+Lkenu7kZfXx8
8Hg98Ph/UajUKCgqqVqtZy7mzsxO9vb3Q6/UoLCxkajCyCFQgGGg8jIyEBWVhbzoSPCdnNzM5599llIpVJkZ2cjPz8
fUqk0yRxYJBIhNzcX+fn5TG020+8zn8+f9KRotvGlUaVRqf9HP/oR7rjjDkgkEqxZSwb/+Mc/Rn3tmVVKluVyuUWN
JpoqMjAwUFBQwkuzw8DBzKaVtNnrSDz30EPh8PrtYlUoly8Xiqq0o/oXIgyQ5pRIwTZ65JGkq5xKhllRidOzkdVN
UVISamhrMmzcPubm5SZMLs6o6enBwYMHsX//ftTW1soz4ezYW5HI58xLR6XRwu+o7++Hw+FAJBI
Bn89HcXEx8w6hoSbKNLnvXxgmavUp8EuxLh2ztRKNRtHd3c2iKAQCAwuFkZ2UKv
VSZwkuVw+oUkakbAp+Zyek/kctS/a2trQ0dExYqJAbwAQb/EI6AJKmWckYPwVImvQGQkbS5CAaDaGbS5SAc2AK7Zt2zb
q37e1tU3p/5JqRCvMVMMvEx1zqQYGG2r9+OJ554OJ554DHggVCoRqQGAAVAgbsUHDGKHyAqsUbQDHwAiwAEACYcsvqM
d0bXhx4wQaqqLuPvk9XYkOGm1Ysu3u17Ycp0tmjUJfIz2WpNu2AqDdHjNkRGPtXOk+sgRkSHwAIg5LoAc2CD2zpwkv
WnY3Lx6yNXxVvgkBYrZHZRYZTRFmZuHu9y8nVe+DJeuNnR7Bb3HbT5C/ntgG+sGCXGHrCSw0hZUwtYGE4QyUOy77Rw2
nG4iKiDK0NDZZcP1HAfVvGADBQUOb0x3BSM60XfV0mR03I9MGHbbrhD2OR3hkxs+akQSOxsmE4p4f/rDw0L0FXjuN
DjBTnUd6tRGwEqrBcN6A0dmvzlwL8wmf8pC5E0VE/7ZqfTqlAABtq0tgrMGkbW99OxB1lxPyKaPgABG/zA5kd9Kd8Kd
Bv2Lyeet6T3J0R/xGKpAFlzEfJ8lhkRT1BsRyCfDkcEDBZ8xHtMYNmoMBvpYAqhLqnxHtvN0gUyjBT9+DCl6OFsPlbQ

ej5eUtUjPvV4vhEIhzGYz6uvr8fTTT+ODDz5gnyGfz8cNN9yAX/ziF9BoNIxbGg6HIZfLWWU2FouxaJ4jR44wdTC
30s11vqZ0ADIlPZEgZXRxcTE2bNgw5dfhekLNFk4K8vUTTzyBuro6vP3220mqtB/96Ec4++yz2QfKVQ+lkq8//PB
D/PWvf8Wnn36apEq76667UFpainfffRf33HMPWltb2Um+ZMkSfPzxx6OeMDPFMRovK+00Tg4olUqUl5eziaFcLmd
8CJ/PB6/Xi/7+fgiFwqTvzfZFPhmIRCImDSYJdSgUYinZxD2gm950DB9CoZAp2Gw223ERvHk8HrKysj4MimbKBa
GJpuJRILxS/r7+6clAmQ6MJ3v62mcBhcU0zSaYSVNftP9nO69qc7j3OdUgScbESKQU+A45b91d3eju7s7iTNGC3i
aTFZUVGDDhg1fTo4RXfwZGRl4+OGH8fDDD6f9vdzc3LSRINzXWLhwIf75z3+O+r82btzIeBY7d4d+7EihUr8NBDD40
5i/7d736XVpX2/vvvT2tfXCAQ4KGHHkqqrNuM+cn+eukWjUeacS5Uabr4UnbwkQSULFD3Sc6rgRKPRJJk8leTtdju
sViuLMiCpPVU4SHbPrWJQXAKtVqm6Q1lgqRcY96ZAF1xqNSldKZZWt3Rxa7VaRqYkhQY3QJe7UeI0N/eI1GUGgwE
5OTmjOhGPBSoPO53OXvzJKlO97nG4/G0lSyycnKgVqsRi8XgwE
Bi8WCSCSCnp6ecY+Lq8ai1zOZTIhEIiz+g/t5pSZ50/7G43FEo1E4HI6k1+d+3gSqKNAkhlzhKBVbC6/UiHA6jv79
/Up8RvbZEIklSV3HPQ5pMCQSCJOsI+ky5r2MymSCRSOD1eqFSqVi7VqvVqvVMgIsVfOoipL6vpLpI6nhuGo9Pp+PzMx
M6PV68Hg8NjmnnC5cqslqtNol0TsRzGvNSqzi9vb3o6uqCx+NJyhvCFfZ/ofcjIyGAChaysLFb5JE808tFJvZ5EIhG
7Tmls4b7XRI73+/1MPs/NueP609Gj3+/H0NAQG/scbv1IrJjQOuN1uuFyuCU+KuSKVVFUWN+ORlKhjqe9IWEL
HQ3mNqb9DVSjuOSaRSKBSqdgkmjs2jHUskUiE5cad7Fi9ejV7v8eyvpgMJqNN2ndWKkcPhgMFgwKZNm3DPPfeM+nt
KpTItS76lpQWHDx/G5Zdfjiiuuu+AK1tbVpVW2EnTt34he/+AU++eQTJBIJnHXWnHdi6deuocQanqirtZMQX/fiAU/8
Yw+Ewk+YPDw8jFAohkUiwlSR5E3300UdYsmQJVCoVa0NOBWSYx50kU/YW13k1k6FdSyTTUFpEiQvr6+Efly5E1D7RC
bzYZdu3bh448/RjAYhFQqZRN9wqpVq/CDH/wAa9euRUZGxbjz/P48MMMPARyzPbjppppsQi8XQ09O
DuXPn4rLLLsOcOcOXPSHg9Ncl0uF1588UU89NBDaeXVF1xwAR555MkZZ2dP6D1Oh4mcp/F4HW09PWhubsann36K115
7bQShyWyQS4sd4dZbb8Uvf/nLCLZtqzhQmei0mEgl4PB6EjzMiClcEyE9AL+Mnx/BwUE4n0vbipjzprctzf4BwUE88sg
jaGpqGvEarUaar/UV9euxerVq2Gz1KW2+9fjyNHjqzNIdXCb/SL+75tjz9BBhmnz6SwE4DDc+xL73HrNF4w/9PBD7w
9hl27dkGr1WLnzp3p3YuHEjfvzjH+H+vHFPcNJyxRzfzVqxfhQH/vLNRXCQbZEhUzEg5b+K+00x/xB3MSt+9tzZWeLfPcqgYkhr5+9twGMYe4aaf0/tD4A3MS
GPZh0lhTE9FDAnjqeSF8bR25/vmT8lIpVL3yxyYAzvI8pPUmIVNDp6N6bnDWOg7Pr9kUzqnUIuRqOnXsfTJ5oEoD
0iohn6PGo5ODjUyxFtiOSsXkO2lDnd7yVJV2V6Hl0lGhr/FSv9elzdwzQUEAvHfWtpeiUTk8+OIfZ9D5pIjsrSVbM/
vvLPsrdrzzFd4eJ9C+h/UfmB6pKSMwh8FoiKrCcs56fZkUDEnnz3cRQq+tv/FTk/VbaUKvMhIC9jkEHfwEIJmCAGaEf
LvKVzSU19pCzmcpsKfEKQ+w++sV/WQ2/kEuMuubp/v6fXmu8XQQWlI4pCw3imJKUrS4sAEnIStjnWjsHD9YmuhZGN
DgXbBrehakWn84ZLAbAUqzbtJEJXbGAT+j5ZDpU6Zzl+DBSnPSpo3IKRi+ZRRMRivFvDBtSKLFMdCEB8FuivH/IDQFD
BjGfv2GUFUB0UEvtsoWwLZQxQ5xy9kmRxnLUp0IXLJF2AeNcLPxIPwZOqF3vdPwH0AncDBL+bMBuhA/a6+nMG3Zn7T6
gKsVqH/oJvyTSaTHGdA2dD0NmOsOx0RTvWN1G0UB/PnMU6A1DGYIE7qbgYRqPGUSXwYJ/fbnKeDUgQzCMpTcnDZkC
YpXpUwYeqeEDMHj6U+YFDzUh11HCGEfiBdABTIOC60qNAY7kWqXtPjURJiVcDONH6aSmvu7qsJPW5jhQA/aKHYSFD/
wbzAHpz79Pq4RbDlnXHOujPyoHmREyWNrXe+tFIP1zrt/FNeuwCDXa/GpKpVA/o49Fnr58rkgbngnqcqu28fSsXG
2/IUMAtwCD6M8fdU+qcDW5NHh5Cx/V1+kdRC/MUR76fQlR3Bf9LkBy1GbNJo0CWZCj2NLUMgbX4OM0MqzJVFo0SfsZ
HO3WehRUH5s27Rfmy/EwOrV8b6XpGuL2tTz1Ot7RBe4eWZtcozf3PZhmlEAAJD7/C2AwdG+0vKsl++R7Ln4a95P/PcC
f8z3pB+/xGpoSAZzoA1Y7JTLoxTAH0ztzUbQY0oUGPgF+gmM/4ypuf8ij2IG4Lp07q1lc4F1NyW4RdcVnoPGRYN5
48+gLKHUBXb/EiNgQmdq0a+9a3u6T/1gtLuYa2hlM9mx5ZtUGuDH+M8NHhmnOX+Z8UpS9Mzl6aq7Ix9yYBm/tT27nr
GT+8Zm59ji9Dw71fjAeR+RqSgFPYNMr0XKNBRTLX6J4nE3vpKnlS18ePONL6rXTn7vqH+Ht1q0Ox0pQdxfsx+XrV
bGnTuxYPi1ujCbQ3pPiKpQWMCxZKbH8yqXAW55/O5VQ0N51lAqjcCCT+kEhjzlBVhiFs65oNvkdcYTFQFuAHtnPe
bttrD0cd0tA+QA1+2+7PHyNOrozI3V5jwDwZxLmYdhHfpPIWQQ5S2JMOXVjb0ZaPJ9Xu0ff7F36xeoDF53mI4FJZSQ
V/p/OaSzjK5N4nW3FH53ffCPWs8OSx7ZGwUjLhRDgXBmVeulzN5Mf0ZOaWRdKN8XIF0Z7jbSoWLwOIafg5wL/hX08
4SQjlbyX1o8mtKFdBTXvuk9CZHs3/b2Pb9/hrbdcuvhqHONx2dTEQqp6t7ggWbeDOP27FsDrz7c+mLt+1aIGVwR0r
Q0tKCf//733jzzTexbdu2UTOqyKCNa46oVCqxcOFCzJkzB3B3w+H5FIBPPnz8eGDRuOu7pDwZNbt27FI488gh07dgA

4prD4yle+gjlz5qC1tRVvv/02Ojs7mZyXBtGJIisrC1deeWWSOoprGFhbW4vdu3cjHo9j+fLluOCCC3DbbbdNuXL
Q29uLK6+8kqWIm81mrF+/HldeeSWGh4dx6aWXIh6Po7OzE+3t7WwjF1huNS4rKwtr1qxBVVUVI9ba7XYcOnQIH3z
wQVIrQCqV4gc/+AF++tOfztpgf7KQrz0eD2644QZs3rwZAHD//ffj3nvvPe5K4slyfCcSM32MPp8Pb775Jh5//HH
8+9//Tvs7YrEYK1euxIYNG/DNb34zbcoBF7FYDF1dXfjTn/6Exx9/PGkBe+aZZ+IXv/gFIpEILrzwwi/k53giPsN
//vOf+MY3voFAIIC5c+fi/fffh9lsnpbXngpmk3x9ykyMvF4vDDh48mPR9o9GIc889F59++inKy8KKTwr1qxBVVUVI9ba7XYcOnQIH3z
jj6KlpQWJRAK5ubCAC0Jt9Ubbd3gxLc6qlSbfI+Jor59KoqpO0MBAqUydBFe0Km5HPaKLySNTfQgpA9AEUqtVCSDEaFPb9LFBRT4ffFZLeyW0TE8QfOqHdj
rZq2NEIlEsGnTJvzjH/9Ac3PzhP5m8eLFu06667B8+XLw+XwIhUJotVpVpIJBJ0dXVh//79+OMf/ziltl95eTlefvn
ltDyTsfDBBx/g61//OoaGhqZsxieVSnHnTnTgDHqnlr1qzB+ee
fj/LycuZsrVKpEAwG0draCrvdjkgo1GM638jtmeOMTjcWzZsgU//elP0djYCLFYjEcffRQ33njjtLw+Hd+KFSv
Q2NgIl8sFn88HsVjMPLHIxkCv10Ov17MgVHK9zsjISGq1x+PxEZ5Ms4lQKIStW7fOyqTh4MGDePTrKSKLqL5NMbVx7ZI7927U1dW
lDXEWi8X45je/iVWrKVkEqlaKxsRHbtm1DY2MjYrEYGZ53tUoWSFrtHX08RZIpUTzFuaGhcZF8ZmW6Su9BMaWvFyaHLDVhXRCOh3YfG6O5p1UYbctNqWRk7TKmrc
/ve9/DSSy+SGEaLLmDZjJ+xsGEaLLmDZmlpKdauXQ
hUIj33nsPv/nNb9Dc3Ixly5bhlltuwXe/+11UVlYmvX4+11UVlYmvX4ikWD+H8PDw7DZbLjvvvuwZcsWAGARDTKZjIXRkmkfQSQ
SQa/XIzs7G0VFRaiursbsbSpUuxcuXKpFTzQCAAu9000u900h8MBu900Pp+P4uJiFBUV4fbbb8ftt9+09957D7/+9a+
xa9cubN++fYR9hVarhcfcfjGUE+njt3Lm677TZcc801Jx2/ZjJ488038dOf/pRNrHycvDPf/4/4T8+bNQ3NzM7xeLzQ
aDTvGREqMDHcLh8Po6+6Db28venp60NvbC5vvNvCxkOpmwCMAxsfNCmW+ZdN7r+Fnqr5N+bvwzSjOvTCQSyEMUKA+JsbjGhDeCnEmEwCnDSlIjaV6sZpd9SYJO4dsyrA
uDkNDQywqQi6XA9RoOVywWazUkRBCUlJSguLkZkZ+fj4L+FuV0X+fo38h4r9+fFiSXuEYmvQMhfqdPq9AqPQZt4jlrWwzXmHHGiAQ38Mos
Y3d2Nbdu24YUXXmcuDOnZHRpPy8CY++7Z3xqahOPH/XO9iQ0sSh1Brk/HDVFQL6RfpxUxMmZuk9j2y6yRiEF7CEFAGxhX2/ru3nj8fBY5n9bQt
/43A4UF9fj+bmZohEIWgGmi12iSDVhLbzBQWL16eK/nMUkeUq7Uc7L
oS6q0u+++G16vd1RVG3CMeFvHIe1h4PM5Mqz0oSf//zno/7fUy1KVu5Tq0fIRH4V6iCwzPAZr9jh/KzOPkC8KY5ceGYnx/9N+y9tKqN/K/VJ
nFrUWSeYlgTSPeKUg2dO8WY/+MjhHM5+O62DT0iFmWb1RGRzZOuYFNN/WLc4+MZdW5eg1/9+cyC6k+jOhTGeQ++VKHV9MuaU1YwC3o7mXK/A8Ey5J3ss+ez2LNVkEeAAfPAj3KPI/vx6lwx+IlUnn6mBRU8rV/MF11qaiPKaUWzUIaSCZ2aOc7cn5Wid8K+kUgyEgA1nLnzwzMGYWxFqCWLsfVHKo/vHD/YPQeYdhPXtRcbyqswNgVH/NvrEnWPkDFVdedjTjg6i0/8xahwUVdPiiUTVD8edGeNq3+xkgAgqWCxGJs0bY9dNIQg1OX6xUVeDgLKR8pJwFhFPDOHxS+sxZmAn66XUh/RP9QnwQA1TW4+0ynuckT3e8dqrMw3sRxSaFEf61RH2h5lkeVTGp5ojj2Y9Xi2d0Nl2zEmIgObMVHsbj9Ap3MBCyPCYQbAHLP4KQeTe/MD8zzoOVELhdRDnzcV9oGyyQRqQwDt8/VwsT2oydcf+iK3fwsMhtMBgyM4pHAAAKzPLkTHRv5HDmQDT9flDyZh3zc2Aq5ELWaqoF7OcY3rAk+GFLlRaE9hiZ2lZC7EvqbRzYpgcx9nAGqs/l0vTXHq2H3Sw8WhlTRUQgQ/8f4FVf5HdZ6MMhcuFK/oT0kd9QEw6yHlwaDvr3hMv3awg5RORpbVtjTwRx45n2D26cpRrSBCIK0jf/xH0lGQnH/o7SHNbfKKg8zgb4zgiGaZHb4L0dJfHUOx+tK+8BcHHdr30d+31Sdp60eQ5HIclkYY9BwGFM3wJqwR1UTEgXIm6Mes3/wHNsXcCzWfxgu2cxRYNJ2UD9dnf5n1V6VfT0Zf35jOG0GaRrm42jsSRjOzBkvgmIIYcn8xU8k5L5gP6BRjkzM8F+YOYfjmdafKZbrlqmMaQ9YRMX4bAc6xTnFk0l+mQWrVwOSGJnA+gJ9mEv7/bbKbU4tnoVMoCoITz5TdDx5+kRWZMSEhDt95LJbv2w0V3DDO3dhUw0rVDt7yb7ReY+RbbeRjp+yjQR+MHiMmHlU/bB4kZEj3HpRl+8vb9G5lxR+2J0lbSv1HDF2hZBiP5WAfpTiBiH1yOSSCe+ljCt/ImNibUvaZrNL5h4o4PzXkvytYCMKpHgS4wClmWbqbeM8Rsgz5ShzqU1O5Oh7XYKjOYzy6Nem90t5fRZtdKdhbP5cdeNjV9XOCRgoeR4CPH4e1YO1cpQn0HCyTPD3t2QcT5NwAyQdr6+o4nCVLGhygvWFH+uuMdAifKUYIYGgbNqZhw9Pdco6aS+2ABtJmPydu/exAq4rZfPSAkDrmPsPZlpCtyEK4q3KNa+xyjTnqbxKWSRcIJfTOsZWrDQcuH2bCt29oJ2PvYbC8VwFjIdXzCZjPbU2q+w4sxpXq3fZaxFZTkF2mPDtdvCKZCECpXPRnfrANuMOpM7oXakmIBE1Oc+cSW+n9ELBgj0WEDU1H4DInF3XgYaapzT5IgBTRi9vjzUi/i5DJ/WqnyBfmw3ZSDJ7J6U3PLfEFUzt1wfmOz9DJBrpQ5TxwkB/hLsXVRkKFyYJRL1i/cCMO1MKwE2Q4ADRyzPmTTyjjr+AeCpBxRpCAS/ixwP5QEnmVMqI4x7jtDxf5zmqJ6IfHvAR+6ZGpSkwZvrEcMzYsw9nq2uuUy5rLlUjAtsEr9mQx5G6yRh3YixZUOmJYH7rjEvUFqt+fFFj3A+Mjv8gdQIDzwo9QNX+SgTSJvxkKD4cWHQcC0qsi26x8L29fKoXdf5XqHX2KSHCpV/Oytib7TdNr9lIbfKLoiTqnIvrf0TEXcwH/fv0sq0OhpJ9SYjVzdlOTNYbiaNW+5Tfq3T6URpVwGhXz1qnqHFCnH1OTdVpxNn1DLVvg7NZ4ARxYEwvvjKEspEqmjcbENAHqrGSm/UqCbbhhNjkgXHBwjdLkSJR9uBjWDMtjn3d5nzT+UgV6QrF/Ts/C2x5r1nl6Q5jTE+h9KpnTGNZwuoxPYg2uaSsXJfrlqJBxklG+p4vXTsvn4odHpCPG2aGR5nYKmbpxfPCFvMA9HWTHpXwwp4MDOaWydFuXrdhjXjq9qWL0FWTLwcftnhMBrQtDsfgazTp5/MTnX/4WlRR6wpA5sb3kHTFeG6KdtlyEtYSRz7/FcTDBChAz35nhxFxk+76H1wMq75OvTpM4XanY3hc9ahmG6nMdyAY+7pYuVaSGyIjB72gX1fnC2Q96S1nzc60EHMaL3dFNb7jSMMuZnRLzB1ZP1tqB7xG2eVMhyLMUbzq5cMdlTJCq8WC9qNaFd8BwqAvsCXhGcwZiOaEEp+B4gr0BbSUIRaTu06n5eDQvItqCuZfhhkdgRXzhSwlKzxe6nU8j3TGwMxEIBfDAUwq4iNg4l93PlzfAD8v56r4o0rFCNTkdWH1w4pX4IqbDSGr8aPB9HpIv8LkSz47lwOmhq1qmtl+rTN0ddsyk7C+1KhzBnGjJYQAp9WC6UH2ecPdG6aJ5V3yVtxYbd1TqW9mPBnmUC22PP9GAVC8AgyK56W9kN3QwElGZ7/R8bNPoDp0W3q8tzDS8Js1lOHQ3I+VE6bYT/P9YPfUtbJmT1eT4pemvnRL+UPO7OmREF3ULEHKT2wGSUyP7Dnp8jOqn1GAdlUKRYk/xRAZyQ6jWNXzjaS/w2jKOQAQAZ3SMFpLTy0dHZwTRbS3M/J5MDIxV8mMHhP3fwVrdAVs5HtqpK9ulUQqgHj7wJXGE5eImTn2CONOwkxpXH/Ig5g7KrCDFBS/eCwgvjx/90pEO25Vqx4sGcFp9BDoGsqPCRrUkKFiaXMcJWlPQhUA0MoqVhJXU5Ln+EQ7Lkek9/30bUWEtlAbSJ5Q8HXS2/L47bH+bQ+7RFFcpe5YdHDnqJRvbJ9R+ZyDLEQSOCVWsYP9+FNvd2uuehbjKmG4xW6LBkJWQ4oTGmw6ZtpRxi6YH/s9zSLBkKUs9/qtSDC0BHsYuM0rNlvAeA5x8mTmkZ3SV2BUm4zCAv4cRqPP9VodQlHXkMoo5L1x3O3I6wchkmEeTfC5TIJTnJU7i6RxDiCd7t/xyWpENFxoH0WV+VuI0h3/Mj1zi2wrjz4FQaSK3qIFHPuWiKAkpj7n1fDLtY5o7gj2CuFaCe2nLbnUK1FPZ+iA8sqmJ1wHoGEF5yNNeoCDl6+ldjeXd5p4CKzQCQ/MwGN/cukgx23MQXWAYIZbbhpMYYIpsVAqtUUdQOTDHMQIJDIJ4oMMkJwmNNF4lKRDVWrTjcV0y7W7j4yXdvtPZS2vTcJPuGRB/PxO1W6aJ3FnRbPIOEhzbtp2qXBPo6BAR3u6Lk1mNSXIRwbEKZiKvYP6N8hW+fZ/N4Y5c/FCg8W7dBd6ohPq6bHIyqZ1WhvDj4ZSiKVO6kZpNi+QugIT6XIf1r/d8nfL/Dd0BEdvLkXh1Gy9eZvKzjdGMjYEV61dsFQ9kXodOgGk8WtnSvr3pmZi/YWN9dxhqTaNMW+8O91wSq4D/DI0LyIlmsqvzXzg3DBe4/Hx0TWJVnMBb6DevJNQxCjfmpAeFKyR/ao54VHy8dAY9WvSWM5XfFvYAe5qOIC+6kFbgiqJm3PXffBaHWvtL8mP5RPyQVPdt2/+B7/fuUpV+SeXDBUSHnbUK8hklnbT+vaEUDTaQrvNx2EuHfbgXx43C1JN8BH/UZ1Ul43k5NCC8wMuEBZAIx5sH/CDbS5gRD0/WbebvWUoMpOzKI5pnvd8ONppRrQDI96uGuHDzZwnDOX/Fhsnmsc9dfOyXjB62Vq/CRK5IIwvmCr4+1TO+BKhM8dOsYB6iDDS8TxklYKc/kOfX68hZ12Ap3QyuiFEGrkA8I9wCcHMdpIiLeVMzb7Xt+1NzPgxnV5fCPBzpgu8rVt9LdSPxWFNXPLS6TNHXAm48t1r4HDV3Lvc2zfyINVSEedqu0TTpXA5YMZlWEBOT8x+K3khoN5xBbnE31RTB9dx3zeBRY7C8pw6QWM3Df7j+K7gBBtfnEDJB/aMdiOnAkA0rgwCZzHgqVCQ6WZAPemdPNuzHQRC0WJnzkULgdUz5zgZcWnOBG5w9YJS0m92gnw4Ak0ub3AwxvE2J0NL9nx9+UHL3ESP3ZhEMHfn6wgoQRCNIA1bnHyGxybVrFeNUrUojZuI4cJWd1aP9ZO7l9RGxjBWkQKPTHGLwflIwOp8L0iVhJBzRFFyaE+f4LhVPBhRNmGd2e5ZoJnGSLqdYKFyQJOrKPSgXUBvmAoWUomYmCmgRwMbU+Cxpf1hFNpXjqtOKmB8lb23dlfQaaHsXMnH9x3iqIhBncXwa2ZchaNGWTx3+1kWLA0BDeP4MNwNvx2sWXRGlSUSDa/rhbj/tgP8SD0q1zr+w8cGHfMxnzUfbLsNlm/71t+wjk6Z+MjgrdmOyGHC/BpwgmAApxu/UUPVbqvV8PdLl1sCAQ6f6GRH8RvK63TOSggnWm2IZtCVHaMnW8AeyLpdQUBHrNjCxgIs+2VuE0FxvwPD+RkGdBu2NW1N6f/wJPXLz2bw3Yz5iNpd5oUnknO/+lTkLUo0J5L/nlMfFlFM5YhBFlOqIpcZPBdLzufGlNXSkFj0hCoEURyG7PCz1ADlhoAd1DJfjEO0yTHIFA+mzxyCbjSkICQWyNIWY0O2xl0VnGfjNQb4iNwxMSgFjrVd5f3NCNPxmEr6uu9TTxQjT6cV/MGmycBRKVEkkfaRNbCgZgQeHjCvd8dj4BQrtnl1tGFiRgj/DTywKnHNc34yc3qY9OfqD2WAyfDLV6XDYrIcuh5OX3CGK7j2WD6+S7QPdYBuLi4Gx3/8SOzD+wqtJD/L+PN4e+qaxNnhWmnE4qJt/gGjKXOUKi4HBE9ZM7UG/fWTfO3b2YtnhLTNfWqE5PS/j0pdtAMqWmO/hdMz8GbGRWk5bVkiHlGoW6rGGzVRb+lRwrrvDhbexEnKdnu2sY7VlR36xbthrL35R9XXuaC1N6bGyekTJJ+BcLm8OvMncEfBvA1vXG/+p02LSjtszoc5G/n9D8z6nDxKF/+cS8Qr0Zb4nMrPaS2JpDiwvVjeEO3g+0vOsV7UnS5qQ7r4d4dq0x6kEcZwJ81jOgY0Dn0UvNhZ2Zbw2aF+x/wXLUBjhHBRRFN7C4OE+IsmHTkw1Up+M1J/Sw3ILf3G3YU5UZ2pSQyNQz+sYmo+3K3QO/Q==

zmpoaCAQCHD16dFSDx1ScqhOjqSCRSGB4eJglsw8ODsLpdEIul0Oj0UCtVjOeTGrlyGazsb8ZGhpi7b/u7m5mPsn
lFRzP5na7Ybfb0dvbO+JmPp3g8XioqKhgsnriMlAAKpHgbTYbKisrkUgk0NzcjKamJlit1hOyTyKRiJE1iW/R29s
7pfaBQCBgLdqFCxfiBz/4AYRCIT766CO89NJLk1rtm81mLFy4EDU1NcjIyEAikcDBgwfxwQcfYGhoKOl3FQoFamp
qWLK9w+GA2+1GVVUVli9fjosvvhglJSWs1XiijedOtuvwROD0MZ76OBHHl0gkYLfbkyZK9NjT08NoAeMtZlLB4/G
Qm5uLYDAIn883oclXaWkpGhoaTqvSpgKHwwGbzQaVSoW1a9fi/vvvx7x58xCNRrFt2zY8+uijOHr0KLKystIq2uR
y+YQnRScS//znP3HTTTchHA5DIBCwihbdDKjKQTeGVLIi8J/ed25uLnJzc5GTk8P6xaSAsFqtrHXkdDrhdrtHbB6
Ph01yToE58wgYDAYUFRWhsLAQeXl5MJvNrLISiUTYajJ1IyKyWq2GTCaDRCKBVCqFVCrF3LlzsWLFinErFqMNVk6
nE01NTSM2i8XCeDakHqJHACMI8vS5+P1+pkikwYoLGoiIJ9PZ2YnOzk7mbisSiVBYWAitVovh4WH09PQktSHq6up
w0003TfkzoHbLO++8k/bnNKHLy8vD3Llz2eSJeC0WiwW9vb3Ytm0bfvOb3yT9rUqlQmFhIQoLC1mrLR6PQyKRICM
jAwsXLsT8+fOh0+nYtcNtGY9nIkiVirq6OlgsFkbgp9auUqlkrvHECaSN+Ekmkwk5OTkjqnmRSIT93WwiEAiw80o
ul0MkEh1XRZlijFpbW9HV1QWxWMzeh4KCAuh0OkQiEca9GhgYYK14WlCQao+7pXuPR9uMRiPKyspm/b09lcHj8Rj
tYPHixaP+XiAQgM1mQygUSvoM4vE47HY73n//ffz2t7+F3+9ni+DRSOsErVaLnJwcxnfLy8ub7sObFE6ZidHAwEC
S1JjI1wDwpz/9CXV1dbjzzjuTVGlcp2wu7HY7gGMOn2MhHccIODbATUe/nBAIBJJya7iYKfLtaJDJZDCZTMjMzIR
Op0MgEIDb7YbX62WVEiJAEzdIr9fDZDLBaDQiMzOTKT7a2tpQVlYGgUCQVEI/nk2lUiEjIwMmk4l5+ZwojPeZ089
Tf0+pVGLx4sVjDjaTAVXzuARZKmdnZWWhsLAwrTqMbvqplZdIJII9e/Zg+/bt+Pzzz/H5558jEAiMmBhnZGQgOzs
bdrudrfqUSiUMBgMWLFiAqqoqOJ1OfPrpp9i/f39Sz60QmkzF5NndCx81xmgi8Xi8OHz48IjeR8NZbb4359yKRKIl
vJ5fL4fV6YbVa4fF4pm0xQNmCfr+f8cB8Ph9kMhkqKytRUFDA2g9qtZpxWObMmQOZTMZaji6XC8PDwzAYDMjJyYF
KpWJO86RKpMkFTTQGBgbQ3t4Ou93OFKbUyhwYGEhbwaRrlzZSfEajUQiFQmRnZyM3Nxd5eXnIycmB0+lEa2sra9l
ON8l5KtBoNFi4cCFKS0thNpuxb98+/OUvf8HQ0BBTzanVaqZGy8vLYwtJrVabxDHk8XgIBAJJJZGgiR9OEQCgUIic
nB4WFhVCr1Unv31gbTUJpwUpEZvoecOxapcljRkYGU5OFw2FYrVZ0dnbi888/RzQaRVZWFiorK6e9izEahELhiEJ
DJBKB0+mEVqvFpZdeiuXLl+OOO+7AoUOHAADV1dXg8/no6emEBy+Vif1ddXY3rrrsOKpWKjWctLS1YunTpqOPpVDG
Z1zklWmnHGwnChcfjwfr166HT6fCvf/1rzBLdTKnS6CaXetMnxRNxUCjWghQ2QLLDs8vlSvLCoSoDcGylTe0vqj6
RjJP4IwqFAjKZjF283OrFaRwDKQUBMAXaFzFXCDhGvA0Gg4zgL5FIJvy34XAYe/fuRTwex6JFi6BQKNiEjnuOkiS
biK9kVknEWtroRuH1etnNXSKRsEleJBKBzWZDe3s7enp6GFcrFAohEAhMiGzOhVarZYowUgn6fD7G/+KSaul5OBx
GIBBgBNsvE3g8HlsIkcKK2qJcKJVKZrtBkxD6jOl95D6me5+536dHu91+ytgk0HEnEolpXWCbTCYYDAbI5XLmd0X
XQDgchlAoHFEN5z4Xi8Xwer1M5CKVShmPKBqNsiojAKYEJirBZNtr42HVqlX4yU9+Mq2v+YVTpd1www1wuVzQarV
49tlnR/z8zjvvxB/+8AcMDAxg06ZNo6rTvF4vNmzYALlcjJEHV88HsfQ0BD6+/uZTwlt1NJ
JJBJs8kdmd6N9zZ0c+v1+uN1uNDU1ob6+Hh0dHcysj8CVVhsMBvh8PqjVaqaGIQM9ejzVJ5qn6nkaDoeTLC+4z8m
sU6PRgM/n4/PPP8gdFF1005eMLhUKor69HW1sb1Go18+3S6XRwOBxoaGjAwMAAm1C73W7YbDa0traioaEB0WgUubm
5MJvNTJJvtVphsVhYhYi8wYiUT5NXgUDAPIRMJtMICxCTyYSsrCzs2rULa9asfAfCfCjh3o0mGQCBAOBxGb28vM8O
0WCzQaDQoKSlBaWkpSkpKUFRUlHbi7Pf74fF4klrGJwKRSAT19fU4dOgQmxz7/X6ce+65yM/PR0dHBxsEHV88tsfQ
2O/r6+pixJ/HyYrEYi0tKJBKQyWRM+EBjg0KhgEQiYQawPT096OzshN/vT/seTkQFlpGRwd4XrrWERCJhQgabzYZ
EIgGhUAi9Xo+8vDymEBjg0KhgEQiYQawPT096OzshN/vT/seTkQFlpGRwd4XrrWERCJhQgabzYZ

OMj2lPC9uZAgXlLNFNxOuIapCoYDb7UZfX1+Swug0xgbZCuh0OiasGRoaYuM0qXtTkTrpIQEAgUQLpaWlMBgMkEq
lTCXm8XiYEpSu42g0Cr/fz65f2iQSCbuH0OfOzY6kjZIByP5kvGoxZWFy8xzHw+rVq/HjH/94wr8/EXzhyNeE6VK
nud1uLF++HEuWLMFzzz036u+dJl9PH77oxwecPsaZQiKRgM1mYwMuVdna29uxf/9+HD58OImMP15I72yDx+Ox9uS
cOXOg1WqTlHterxcajQZGoxEqlYqZYVKFBzh2A3O73Sy7biZhNptZ9E1JSQmTuvv9fgSDQdaejjkajrIKnUqkgFos
Zf46Cj7kbTTRH22hRmEgk0N/fj0OHDqGxsRG9vb3o7+9HNBrF2rVrUVBQAIPBAIVCgVgsxjL+uJSCSCTCuItcsnr
q81TfHm6unMvlgsfjgc/nSwrt5krwpwukIpttOxcuyANOLBbj/2vvvcPbus/r8YO9ARKTAPfeErUlS7S7ItW5ZHLDv
xrL+uEydpE6dJGjdtRpNf27RPk8Ztk6ZJs+PR1PGIndiOt+VYsmTJliyKEkWKew+ABIhBEgAx7+8P8P+f3kggQpkiI
lkr7nee4DEry8uB/c9X7e97znxGIxBIPBGTYt1KxCZHjKdlOnNFUYduzYgU996lMC+Xou3H///SndaCTK2NHRAbv
dzmYtn/rUp3D33D33XfP6EYDgF/84hd44fj8nJSWzatAm//nQ39/PeEh8XheNa7W
PcT643GN0OBwpv9tsNlRlRUVOCmm25KeZ+yP9P9M7Fl0uFyvkNkAGtWCxGKBRCV1cXtm/fzgRC7XY6ZDZrteenh6
8++67aGlpV1p5LfldrvR0NCQotLOCrzzjRV2dqYdr9Ppppk5x0OKKrovLOcrzzjRV2dqYdr9PpRGtr67y/HxLMo+DDZrPBBbrejpKQEJSUlaG9vx1XhzeVr9wVx
SbaXVaqFUKhEMBuF2uzE1NZWiSE7yCl6vFyaSEvLw+BQBuCrc0uUdKGhvnYhBCtVmTJNRQ9V1eTa2yiBiIpeDEsdpydevxvaDaUGN7tuFh2rFvwcHo28JTOE
qNTiOY9UGKg+q1WomMiuTyRiWIx5n5A6803QyqQr+eBG264AXq9HoFAAI5S/ATWpAMXXq9HoFAAI899hiA82TO3t5e7N6l6k
bHz9+HDKZDK+//jo/+//nPo7W1FX6/H6H0FQCDfccAOi3CdfccAOi0Sjeffdd28vk7SPRCK
QSCRQKpWsHDC9hCgSidhjeUKhEra2NvaVKEEFfh80Nx2b3rPiwQS
sw28+3WCX4oFKOj/8UmEkEkFXVxeampoQCoXYA4f87fx+P0ZHRxEMBjE1NcXETdVqNevUNBgMyMzz06z
nGPF06urq0o6RHoIClg7UgZbOgYGwEHFaChoUCgUqKipQUVExpyH65YJMLlSQ9lLLjVUTGJEORiQSS9u/
FRHv37sW9996Lf/mXf8/xdO/PCHP2TdaY8++ighjY8++igjYRMRELj8N4/3qqxBBjHiGr8lGbmv/J/phs
baZTk5eUxZ2SLxcKibYrQ1UgsBJ8lJrQmtq86+ylQ91BiuG9/+DEZmTSx8GpWZkxpRjXFXDHtaoYIxnU2cEFvVEOYxmsnkzCanKZmwDaMu+kyFrm44Csd3Nd8WNOzVzLYrNrbwXIpY7ssgTX/Oc9/YWjUrwdyDwhtxqH0bvwhsl5yVlQMMPYQMP3aQQ
0PlkXEP8k3XokSEZ8kEsh8kZq3jKZjDndx+PxGbX24eFhnD59esb/q1Qq5OTkwGg1R2HoIkZhbZDVJYuRJ/+V1jlxbuYG
qlXC73Swg7O/vZ22yubm5pXeHpqYmBAIBJIw+wWsBmXJZLOO4ZQ/cYgDVKz0t9P4nKUEajBSb4hSBxxjuFc/lUGBHKnlR
aJM8Xw6MOtFgshoGBAUxOTkIaHgmwTNP11OtcvIyMDVXV1LKsVCoWdVxzVq
cDkeKmCoA5Ofn45prrrnOfRcgQMDFYdUERgbtSCbaaRBnpkDT7q4Fng/zf
a9S9VV5rX60Vzc/OSbS8d6Oa9UNLqagS5vefn5yMvL9HdFqlNbLj0sIeNsyjgmhcXBpQ2cF4eYRTHWnBFvlBTU0Nq
OKFPlcrlw8OBB7NixAw/E6OnKYkL7Y4k2g1tKdMOZmSdgaUnZPlsGmhQY9Ozh77mZuV4h8PLo8OI3bPWt1l2yOP
0v79/VksbPiQSCbKzs1lKPjpsl9ZLy6P+KF++/fR6LZ1JCNlYU78X0zxlqCspjmF6MGD9Xch0tiGGS1p3y+gHl6
fGE0/Dw8Pz1BsXgx0Oh2mpqYueaKtt41KRYx/NXMe7EQIn/QUrVGR0X7Vjr9L0VMFgYz5ZCqrkDvqspYTtvgM6gOvVD
JJ59kk814PA61Wg2tVguHwx4rSgoGUFQBsAJ0UCgGZc6AkTeBIgwP8JQaBMbHGOF4mMhZaqnhTRQ4t9lzcMbk7VVMJYvDuCcm
zkcp107OfZzOMTtctRvc1o9GIsrIytutVv+E5nO3pGrg6uQDuKYGL27cYKWkzXQMZEz3kqZ/EsUcG5/Mnk2ahR0QBUt60r/n
wCfXTfd/MZjNKSkqYplnFhqVcC8HJMKjPFAS6UoFQ8RIZA4tDfWnpO5R/Z2ZtZ46dL2EGKyDw+uWtDXCpeF/I4yZOjQ8d4xL
hRZQ0lxKCJcgFsJTdaIcOHcKePXuXuYr8tcuFRdaX6/n10cfKn46R5I6dpZ6cFAwozUCul2u2c9EjKgk92U6vVrAS
UrttGKpVCJBKxjiJ6iESj3S9c6IdLrp00Ktv/wbH73Sz+kk//kPsXQ3R47jWHlMo9GwMtRt8
xeTOl47LQQr/T9H0tHtShQtpV1J1iNBqRl5cHg8Vcl5cHg8Vcl5cHg8Vcl5cHg8Vcl5cHg8V
nQ1ZWFpM94Gf2pv/Mf1j7fD4MDg6m1IGp1WrCKAFosFS0jKhpKhoXp2Jq03zbK1CLBajoKxi
cWXWIRCJmSE5SKYlEpAmvktQAx3EzOkFJDZyUwacvkUiENQSFqSNVHGZmFAoFAoE
JQeeOHTvwuc99Dt9884033zzzDOMFAJzwB7EnSZG6Yf+c9nh4eFLNRFeHLP3J0PF5cPp3tj5wZtjTgl5V4L0a6
vf//7KZmj6VjNXWkcxyEcDjOVTrqf29VkzqF29VkI308VgamoKZ8+exejR
Z6PV6Zli5i5ODthUgqObVyQSgwbKDqqObVyQSgUwmg0qlEgJ4eHhwOB44ePYwOYPh4MLmpy8ePPuRYkXWPdXL
Tg5ycnERvby/a2vh9/uZ7Q7xll5vimnPyOJQo8/H4/IxRT6SwG9vR1dXV0Ih8MzkFEzTMVSpVGhra8Pp06eZq4PjoikWx83
dd98Ni8Wy6OMRDofR19e3bduHo3g8ABAhHdHBHyCycVGjZu9PPTB2vWDqqPfRS5Ymyp66thDTTCMZSBaCtLBXw3MOGIn
WdUjRJIckC1paWpCTCfk8vqQXKlkyaWZqQV4a1PVA2lpjtKIAB9TTaY15CWGCGJZVqQhR
kYoG7xWJhE0x+aX3679OV7/mTCZIcGB0dRXt7+6w9Lz0mrNVerSz+/rMOgMOs5+ak33IBLl0G
mV6lUikQikTJJicViaeVVVSB+ss7OTVR6kUikCgqQn58Pw4HHY5LkqJbSaN6qa5CMRGbsNKc9hPHvtl2A
7fv3rXwtdaReC1WrFmTNdNUt4jF+QvfelLyM4bD4+YeN44cvmrvmM3K1mx+4+6wpRY2emE
SlkINmMpENOMnXtFsSCQSGBkZYy0R4gsU4aNN7EgbrSvyDsYz3bMkYDYKDkqxSelh4fOhyh1dVCa
U3+9nhqLd3d0IBDhsyR0kBXvNcw4ghMZWxd3wE4e8lXt3JjZpnkJIXmqU2jh2d5smUrBmfYv44aG
4WTEKxIjsTdm5UCiUYgkSQn5+POpbMvxwMs6Ak9ZlAlbYGjM95Bv4gkzfvxoZN8kFgOz
WqzXt+zKZDHV1airq6ivr8enPvUp/PSnP015bMlubzAY0d2dvk+T2q0SIRrEKpVqtqI8+lYW6DD/6IOYxJQ
1mJYC0WgUr7766dv1GGhuPYeOkJsDhMbCyY2YhBb+XwjoqGWNvoYyK5KFGhUsIRjWnxUMkyfgXvf8mrtTc8vhc6tw==
KhQJGoxF6vZ59Ft3TAoEAXC4XJicnYbVaGQGdRHmplDr9PrwcXb5rsitNLBanZfSbTCZ4vV4cPHgw5YZZXV2N22+
/nf3+mc98BsD5UhoAfPOb38Tu3bvR29t7wYfBWkQ8HofH48Hw8DD6+vowNDTEZlFEZu7v70dnZ+cMYS4yruSXE+j
1QsHDhSQS0oECJUoB89O7lxJmsxmbNm1CTU0NbDYbMjGZUF8YmICgUAAra2tMBgMjL/i9/sxNDTEZlFEZu7v7
v6X7PByqVCtu3b0dNQ2Ki4vhcDhgNBqhUqmYJcjIyAhcLhcGBgaQCah0+kQi8Xgh0+kQi8Xgh0+kQi8X

ZjmZmZrKW49keDJFIJIULRm3KtEilUgQCAZYBVqvVLBshYHmg0WgYh2elY6V2fVJZa6nhcDhSnndrAZdTfX7VBEb
pQKlQnU6HPXv24F/+5V+wbt06xONxHDhwgHWjAYDL5YLL5WI6IT/84Q/hcDjmLX2/nHjppZfwt3/7twgEAiwLlu7
holAomPovWS7QazQaRUZGRkqaWqVSIRQKMeNA0ichifyFgC7odN1X08EnaxJhc2pqCt3d3VAoFCwAo/Q/RfJ8V28
+LkR252dMSJ7faDQiPz8fxcXFyMjIYEEV/7tIt0SjUTYLInPEyspKbN68Gbm5uXPecOeaiY+Pj7NSldvths/nQzA
YZPwtmoVRQDA+Ps64RlTrJ8E0WkKhEEvl84XeNBoNrFYrrFYr60IbHBxkfC6ZTIaSkhIYjUYEg0EMDAxgbGwMBw8
exMGDB+c8toSf/exn81pvLqhUKhQVFcFmszGSN19iYS5QpnE6bDYbampqkJWVxXhW6ZaMjIwZJZ9kMgmFQoHu7m6
88sorLAgcGhpibva0ECeDP0tWKpUwBwAjSO4zAxMbEkvlkC/gTSgCLSswABS4FVExi5XK4UjgaRrwHgv/7
rv9DQ0DBrNxpw/kbOJ1I/88wzAIDXXnsN999/f9rPvFFRdaePj42hvb7/o7fh8PvT09Mx7fbFYDKvVvynyl6IFBlgm
kwZObm8sIzRMTE6wbiG9qS6/EI5h+k5ovx4jjuBl1ez5HanqrNPFCLsUM8UIZqrm6KFQq1bLOtilVTqno6SBuxdT
UFAwGQ0rbPMdxaG1tbPMxvV6kJ+5/f8NrRPLwmJ+Sf3Q9IRzzKvvWdIqbPjxDwSG1pCvvsOj8W5ww5ZWmIFSV0MfvtlzPV+l
VIwk7t1Njsv7MZfAzAJBYcFAPvu7nrhBaQyDGnMjTAuTz7lCyczeB+Ln/o6IujmjeOGQ6Gip6YDhpJ0KV1pLOWWVWHHkZ
/YjCUvUx3QaQI9D2QZVc+Xp0uOUjy/tFWcXhHydAmqsOMS4O4vk2jXWPAbHX80oNENJDo4TkYRXHqjZH+gYLHfAzF8J
V7WEYTt/GcgVfBhFSmhUyYzOCXNi07fYE4rS6DXWlJttO93XpLRDKiK3uT0NrOrnp1xJEcK7ICdp11YH+9jSUdVrOsY
kQlfUKUZrTXpvdUlBNMYmy/Elfa1GPHzGPWOc3M3hbDH0e9u5H93M0xOJdn3bdgwLWxmJeyOR94bL8GTZrmbbRfjb+4
4FDD+lJbyDmzfRyThDcJKbzUhyCSqDMZT4vywYmMe6fOk4IFhfWFN6PXdd0I/KB8fsiZRz2lXJrHWRGZW9t20Lju1QCN
Et/Zh8HTa/nRphmvbmyaqcUj/n5HKzxm+pvpKX8GB9mjZ5WHNmHC7fLvOnJfWZtYc9tpjgyqtUp9+QwDaDJqD+Y97Gp
//Ib/KxMz4vSuetWfcc6HPdRcz0uZbkW5+FiDf/Tk5BJkp+tvjFgEwrhhu2zK4fVdTeccxzqz9PxDYqU3aNfGK/0A6fYZ
8W/7wN6N7LVWsrhbl/MZ8A8HTJfrYJLpFk4lmLCMkpVEb+AZvTYCUmWnwHJ+J7ZfDc1ZRkiSvBxJm3RE16gbWUgUhWk
43OKB6GqhkJwUvUxP3Aug5XJjgGiWxF86d+WsY1ouSQCPPlWFiVwm8fWy2HnhMxQ8zK6ABn5N9tmq33v0CERuEyLkC
yHo9EKsqPNhzo9V3LAUmDxBmm+6DJ70X5bbXAMwQAFcGOIxaNnLRK6w2mqgaBXsqdMRsDjhU5Lbcg8MKzCP2RgKBnk
5UEc5ZbdB1v5wWQF0gJEChKxHsWmgAlF1NjYpTDZpY3u/jaOsFLdYzdsSyqWLtM/4YgQtAwIiRMhM7DOqPswJiEQW3Ox
9XTVOwiRf3qWmxmHdQ93fHVKQ0t5/XtC3p63RqEOKxHSv/t/3Lmib3/lQIWqYe5aZPlTV3JMfyxvYLmYUoepSNyPZIJ
4QKPjHISgDOGYUhl4iCjWRLw0JtjU1UTuvyMzYqfdf0evnGkBvyDj6c0x3TseaPB2rM3SqYX4fc5YZuBWaqPy7BkjxyZ
MYqFy+BFMvEoywGX8ZXffUEnCxMUsazIkSxAI2EE/sVFmbWm+aydB8PMZUzNKO7SlYH14MXAIp/jy2kQdgcdH+SfRAiI
cRTKnxl4rD2aByL1yCsaEk0y2dX6W4FjVWlbwXlJ5XZcqd9VjHwH9HL4knpIEZ2SKh9ubC4Ma5cLUtGUKN3g40IRukLg/5Y
ZnRd0XaZbRdyJjvfaUWyCRYKOeYjpRYj0B7Ivh9EvHCKr/22UGNZKfl5IPHLt6ZKl/wBJIyrddaUQSbS4iDPbF3Y6X6/+
cQPkS27JpbMkNPoSZr2V6o0w0MZ09fqzl1uHrTFjJbh5LjIm1PxNZcOhM6pcN9yBZwZ4SR4CqHk6gNL+d7lpSHd+IXQ/
AwApjqAvGi5ewsE++yYAmTDe5CBOPYyiBDB2V5qVNHxKFDm6EW84zuSNqnPN0C+s86IL7eL/59NFaEkfBcfBr/V4+cI5
nKVCKEAfcrXwK7lPlwHQtZ9KcgmcmtIVQJjrm7jfxl2IM4+UJyu+8V8TTN8IDOxWzTvRwqSvb+NmAL/7zgaHJ5aWD
QG2yFTpFbfPI9sqP8nPdTGlTV7rx40Z5ppHs/kB3Uhq1FYOWP3iL1QDNfUmY/hbq8sdSgxoLPT0YP/WkDcCoyZyHDo4
Kx8HL4IxyYL/95kvUGMYWsNHZvWQdTQhhoRl3Q7LNTjS+pOFsJwnXb+dmVR7Er5JxWeRN4yjN7yV5UVV5lP3T+WlySkQ+
4gT0VRQmFktx5ATKVVR46EcUGBUnxV5AhKyaZT7U/bqlWPa/BvPyB/4hPf5QG1I2YpnvpaZvWKPdOgPiwdg6t4N8YeA9
TZSPBVS0m+qjmGCDXNfeHoDK+CvQgnvQmVNfE2jEmmeabpP1NnHO/TcqH+AEoYSfuUJ3FY7nX2WVgaTmnYaXEq87b0A
ZPKPyU0HBbNDBmNDSPpxLxOmzVRiqZOttGOtZZsdhzPkCpSfmvhbDfq1I/EZeggWPJ94crmEHlV7KkpsJqI7hhRyBJgJ
mYPZITb1RyRd82GDf7ki1O/N7TbLsB6yrc/x6mAUGkSHIuSwsU7rSlnoSNcP+N2GeT9NfKl08yD/Gl6VkOEB/YXkSGNq
zO0kxM6DZ2vUWt/YRGEqANUHkiKSm1Km2qdOcwFubsVgg6HwMEd5E+6W5hjpDLhIYVMfUD7fYxyvL7n5eX+F8GTwWrfG
WV/QHbALqmj2FPY+pEDhHNKNhRQ07oomC+XpEuZmEYYlvCRgDhiBoTxCU9b6DzNGQujI0jbCdvGYZ+v4hmDx/MPXGa92
W6RkCKdsf4cybVFIKTjU3tC1j7Jo6ywGaTQh6ESZRaNcrP7FzabNAfTtiSFAnXgCUPAQHVKHuWSWJ9FGL+IC2X5QYa5d
/8u4ZuDBNEQSZRmPQNJUqKpw9hDBF5kDhdZgX4ZJfNBrXmBCXYDnKUpPyYhndwcRXUyRkcMc7TOYLlnDHoT2bH5Jxr/i
ISjZoOybEQfNNB0dRXi6CDEthNqIaySkSiOdqSKGk+7ifxXK3xW93PAYPVrhcbx3mHpKkeSfWj5r6FYAh+jb88EiaF0

VV1mb7euvvz5rYLSaydfEn1lpWM2k3fniUo+RWs7JtTvd36uqqmb9/6ysLNxyyy245ZZbLvhZHxZS686dO+c9Psr
AEnJzc7F169ZZ11+MltVyqPPPdp5SyWyxWKp9tVgsKC0tnff669atYz9/WM7TPXv2LMn4qDRKkxfiiJLmGF8/irL
YJP+iVCoRi8UQDodZUD29/B4KhZhUTH5+PgoLC9mEhpwTOI5jAfSWLVsES5DFwmg04vrrr8ePf/xj/PVf/zUA4IY
bbsCjjz4K4HzWxGAwwGKxwOfzYefOndizZw9effVV+Hw+3HDDDSgoKLiMIxCwmuD1evHqq68yIUSyxiBQqY5mkuQ
NVFxcjLKyMtTW1qKuri7lITo+Ps5me/xMjVqtRl1dHWpra5mRKHWYzEXw5TgObW1tePXVV/HSSy+hu7sbpaWlqK6
uRlVVFSorKyGVSuH3+1lDQ25uLjIzM5f9+1st4DgOPT09aGpqYqa0kUgECoUCGo0GFosFWVlZqKioQEVFBRKJBAK
BAGQyGevOSXd8KDl/OQnFk5OT6OnpwdDQECYnJ4XjLgDA+Uwj8a2WAxzHLfi8FyxBLgI/+clPcMUVV2Dr1q2wWq2
QSCTw+XwpliBdXV24//77kZGRgX/6p3/CuXPn8Ld/+7fYsGED7rrrrss9BJw6dQpPPPEEGhoa8Itf/IIptKrVauT
n56OgoACFhYWMXEidJwJmB6n98i04iEsjl8uRnZ2NvLw8ZGdnz+jASSaTjMPj8Xjw+uuv4w9/+AMOHz68IFJ1OhV
yCkZEIhHC4TC8Xu+Cx0ZO91arNaWDMZlM4vTp0/B4PCnr9/b2zsmlA4CtW7di//792L17NzZu3AidTjdjnUgkgjN
nzqC1tRWZmZmML9DV1YVTp06hp6eHdc7Z7XY4HA5WGqOFspjxeJxxYPjw+Xxoampi1iBkoxKJRGC1WpGdnY1t27Z
hy5YtGB4exunTp9HW1obOzk5EIhGUlZWhuLiYfR+0UFaNX4aKx+OIRCJwuVzss/r6+tDc3JwiArlQaLVabN26FVV
VVejr60NbWxs8Hg/8fj8yMjKwceNG1NbWMp2iqqoqVFVVMQmBWCyGo0eP4uTJk0z5uqCgALW1tdDpdIhGo/D7/cz
xnsw+fT4fRkdH2UKqySqVinXX8Usmn//856FUKllHKJVsiCxPnWparRYFBQUoLi7Ghg0bsGXLFmi1WgQCACAZw5cwZ
vvvkmGhsbWTYhOzsbpaWlKC0tRUlJCcrKylBeXj6jS5bKmjRmr9fLSkXUsq3RaJBMJplwLYnX0s9UjiLS9VwTBnr
QEk9zMQgEAujs7GQE/Q0bNsz5uQLOY7V9P6uGY+T3+/H888+n/bvT6cS3v/1tPPbYYwiHw7YwiHw7Db7SldaQMDA6ioqEA
ymWREVaPRiG9+85t48MEHZ/3cldqVRl0GiUQCYrEYWq0WGRkZqKqqwqZNm7BhwwZUVVXNmjqMx+Po7+9HR0cHhhoa
G4Pf7EQqFmB0AcTn8fj/6+vowMjKC8fFxTE1NITs7G4WFhTAajYyUzW+P5xOapyudkqgecUlyc3PZdvhcEsqIjIy
MoL+/H+FwmPEwiEzK506RGBzV0olfMh9SsUgkQlZWFuRyeQoZfDZUV1dj3759qKioQF5eXkpQFY/HcfLkSezcuRM
ikYiprHd1daG9vR0NDQ2MC8KHxWJhga/FYoFSqYTX68Xp06fR2trKbDDmC6VSiSuuuAI33XQT1q1q9fj66uLrS0tKC
lpQVtbW0QiUQwGAxIJBLsITr9O8nNzUVBQQGMRiOi0ShGR0fR2NiY1stusZBIJLBYLNDr9ZBIJMwiYSVAJpOhtrY
WFRUVKC4uhkajQTQaxcTEBDweDwYHB9HS0oLBwUEAYIa6iyWykz2P2WxmIoDLhYvd18UiKysrRbl8qTtlHQ4Htm3
bxjKgJ0+eRDKZRF9fH/r6+pgQo1gsRmlpKdatW8eWgoICZGRkQCwWw+l0MkuW4eFh9trS0sK8NvnIzc3Fxo0bUVl
ZidzcXHb/nJqaStELUigUbFI7MjLCujl7e3uh0WiQlZUFg8GQ0jii1WpRVlaGiooKiEQi+Hw+nDlzBocOHcK5c+d
dgs9lgMplQW1uLTZs2wWg0MgFTj8fDAl2pVAq73Y6CggLhe3uh0WiQlZUFg8GQ0jii1WpRVlaGiooKiEQi+Hw+nDlzBocOHcK5c+d
Mk28TAEYEJsXbuSCXy1kr+9atWy+opXUheL1e+P1+VhKzWq0XFBtMJBIIh8MsHc2/6fGzIhzHweFwoLi4eEE3Eq/
Xi/r6etTX16Orqwtut3vWdWkWLxKJmNw/BXbZ2dmYnJxkganX62VNALTMp+uJSlUkBmc2myGXy5kQ3blz59DX1we
DwYCioiLk5ubCbrdDJpMx1WMg1eeKuiL5nltE4qUxEC/PZrMhLy9vXt/hdG2iSCQCp9OJtrY2OJtrY2OJ1OluUyGo3QaDT
w+Xzo6upiE5KxsTH09fXNyFDp9XrU1NSw84PkIOLxeAqxntr8VSoV07OKx+PMEoh8znJzc5nn4eTkJCQSCQwGA6L
RKFNU5hN2+a3xlBkcGhpCZ2cnuru7kUgkoFarYbFYsG7dOHdlRUVECtVkMkEsHtdmN4eJgFGENDQ/MK9KRSKWvNJ2s
WPkhMle8LR40HLpfrknXU0T1FKpWit7d3VXbykQQDP5tLCuHp/PdWUhv/YrHmutLmi9msQzo6OlBVVYXNmzfj2LF
j+Ld/+zd84xvfwPr166Duu++m3d5K7EqbnJxEb28vu7mRaKPb7UZjYyNOnTqFc+fOYWlLy9vXt/P1+VhKzWq0XFBtMJBIIh8MsHc2/6fGzIhzHweFwoLi4eEE3Eq/Xi/r6etTX16Orqwtut3vWdWkWLxKJmNw/BXbZ2dmYnJxkganX62VNALTMp+uJSlUkBmc2myGXy5kQ3blz59DX1we
AcQOpFExl4YKCAmzcuDGlqSUYDOLEiRNoampCS0sLRkdHmbTK9OCCSqAcxzGCe0lJCYqKihAKheB0OMcEyauF
wGD6fD62trWhvb4dUKkVGRgYKCwuaa9cuhMNhVFZWwuv1svs+iekSF46OAwkbU4l6oTprarU65f6u1+tZJyjJkZC
6tc1mQ11dHfbu3YtrW+JdJySySTGxsZYNykZTe/atQuf/exnhYzRUqCkpAQTExNprUOKiopw3XXX4bOf/SyuvfZaXHPNNbjuuuvwuc99Dtdffz2uvvpqXHXVVbjyyitxxRVX4PLLL8dll12GSy+9FJdccgkuvvhiXHTRRbjwwgtxwQUX4Pzzz8d5552Hc889F+eccw7OPvtsnHXWWTjzzDNx+umn47TTTsOpp56KU045BSeffDJOOukknHjiiTjhhBNw/PHH47jjjsOxxx6LY445BkcffTSOOuooHHnkkTjiiCNw+OGH47DDDsOhhx6KQw45BAcffDAOOuggHHjggTjggANw/fXX47rrrsO1116La665BldffTWuuOMO3HTTTbj55ptxyy234NZbb8Vtt92G22+/HXfccQfuvPNO3HXXXbj77rtxzz334N5778V9992H+++/Hw888AAefPBBPPTQQ3j44YfxyCOP4NFHH8Vjjz2Gxx9/HE888QSefPJJPPXUU3j66afxzDPP4Nlnn8Vzzz2H559/Hi+88AJefPFFvPTSS3j55Zfxyiuv4NVXX8Vrr72G119/HW+88QbefPNNvPXWW3j77bfxzjvv4N1338V7772H999/Hx988AE+/PBDfPTRR/j444/xySef4NNPP8Vnn32Gzz//HF988QW+/PJLfPXVV/jmm2/w7bff4rvvvsP333+PH374AT/++CN++uknfPrpp/jss8/w+eef44svvsCXX36Jr776Cl9//TW++eYbfPvtt/juu+/w/fff44cffsCPP/6In376CT/99BN++eUX/Prrr/jtt9/wu+++w/fff48ffvgBP/74I3766Sf8/PPP+OWXX/Drr7/it99+w++++w7ff/89fvjhB/z444/46aef8PPPP+OXX37Br7/+it9++w2/++47fP/99/jhhx/w448/4qeffsLPP/+MX375Bb/++it+++03/O677/D9998jDAMDuD

06dNsmQ0KhQKFhYUoKipCcXExioqK5nSkuBRYcxmjdNYhAPD+++9j27Zt2LFjB/73f/8X+/fvR2dnJx577DHce++
9abd3qUppzzzzzKz7wAfV9EmhmmT7SWW0rq4Ou3btYkS9lYS1UmaaCx/GMSYSCTQ2NuLIkSNobW1FWVkZNm/ejLK
yMlgslpTsDf+c5DiOKeGS8KTP50NjYyPOnDmDM2fOoLGxcVlJyMCfNIj4woEcx2Hz5s2orq6G0WiEUqlENBpN6Ya
ijrjh4WEA50tSAGY8OHNycrB//36UlZVBr9cjEAigo6MDHo+HBQfBYJBxB2UyGcrKyvAXf/EXy+ZJttjzlOM4uFw
utLe3IxKJQCaTISMjA8XFxSn3Q5fLhWPHjuHFF1/Eiy++uOBjSOU2KtmQaOZ8gpLs7Gzk5ubCbDajp6cHU1NTCAQ
Ccyq8W61WKJVKli2ZC2azGbt27UJlZSVycnIwMDDAqg78fZRIJMjIyGA2HHq9HiMjI2hqakIwGERubi4KCwtRV1e
Hmpoadj4NDAyw7KJIJEIwGGSk8aWATCZj8iHULVhVVYUNGzYsulNvLoyOjuKtt95CW1sbs7wpLi5GRUUFs0BJl80
FBPL1koE4Rvy2UCJfP/XUU/ja174GnU6Hzs5OSCQS7NixA2+++eas27tU5OvJyUl4PB4mZEVdBcPDw2htbUVbWxs
GBwfnTfKzWq3YvXs3tm/fjvz8/LSGkASaWUSjURiNxmW5OATMD4vR+kiHRCKBpqYmvPPOOxgaGoLJZILVaoXFYmF
db3xbCMpQGgyGWWfrRNBubm5mHkxEsp1Nb4Ravml96j4kwu98IJPJkJeXB5vNxlSvScE3OzsbVVVVKC0tRU5ODnQ
6HXw+HyYmJqBSqaDRaNhNNxaLsfOclJeJaHox3/nk5CRrlwf+RJQnH7OFZnbWGmKxGLq6utDa2oqenh5mjUOEb74
H3nygVCqRkZEBuVyeoujv9/vnbBogyOVyxo+JRqNpgy29Xg+73Q6z2Zyi+k6dmpeyLJhIJNDX14fW1lb4fD74/X4
olUrk5+fDYrGwbtzOzk50dXUx3o5EIoFSqWSTF5KlmK2rVK1Wo7a2FlVVVSgvL0/LJVvt+NCSr2ezDonH49i8eTP
eeOMNrF+/HgBw9dVXo66uDj/4wQ9m3d5KIl/H43FmdRAMBlmHGS1utxvHjx/He++9l/LQkUgkyMvLYzoeRAzkGzg
SiHhJ3QmxWAxjY2PgOA5VVVVYv3499uzZg7q6ugXfHD5M2ZTdu3fD6XSir6+PWSOQO7tSqWS2C4ODgzh58iQaGxs
RDAaZtg8RM202GzNznW5PQMHNxMQEgsEgZDIZRCIRTpw4gUOHDi1KH0kqlaK4uBiVlZWwWq1QKBQYGRlh3WqzQa/
XY+fOnaipqUFbWxsaGhowODg4g+g7G0iiQavVoqamBuvXr8e6deuwfv16lJeXX3J7lQ/LebqSxphIJJjxKWmITUx
MwO/3QyaTMW2sdPpaBL/fz7g/TqcTg4OD2LdvH7OYoQ49fqA6Pj6Orq4uVo6zWCyrQvRyMccwmUwyblN7ezva2tr
Q3t6O06dPz7hfiMViFBcXo6amBrW1taitrcXGjRuRk5NzyQJ9IWO0RJitK+0nP/kJ/uqv/mrG+2ReR4KKF8JqIF+
HQiG8/PLLeOKJJ3D48OF5PyDJJHM+yMrKYgJ1pHtEi1QqhdfrTZEYCAaDjG9RU1ODsrIybNy4Edu3b1+erxf
PPfccDhw4gL6+PkbeVyqV8Hg88Pl8l3kPz5Nz77jjDlx55ZVwuVzo6+tDb28v04UCwHzMwuEwXC7XnJ0qIpEIW7Z
swd69e+H1erFhwwY4HA5UV1cjPz9/RqAciUQwODiIqakp6HS6FPdyvoCfSqVaUXoqK5VcvpQQxrj6sZTjSyQSaGh
owJtvvon33nsP7733Gk2GRtKua7PZsGXLFmzZsgUlJSXQarVQKBTgOA5isRhRFRUUoLCxcksaepT6GH2rytdVqndG
VplQq8fjjj+PGG29ETU0NEokEPv3pTyMWi+HgwYNrLOO+9kRMje3t4ZBo58J/qMjAyIRCIMDQ2
ho6MDPp8P4+PjkMlkMJlMifjaGpqwvHjx/Hmm2/C5XKlFSqcD3p7e1N+JwJkQUEBqqurUV5ezh6kIyMjGBgYQDQ
ahUajgV6vR1ZWVspisVjmLP9NTk6m2Gw4HI4FB2Icx6G5uRmvv/46Xn/9dDw6dOiCQaRGo0F+fj5yc3MZoTQWiyE
UCjHSosViwaZNm5gwm0wmw/j4ODMFJcNOfvsuBRckkKfX61lWNBqNoqqqCnv37sWWLVsWLVsWVBZNJBKsdHvu3Dn4/X5
EIhFotVps2bIWqZNm5gwm0wmw/j4ODMFJcNOfvsuBRckkKfX61lWNBqNoqqqCnv37sWWLVsWVBZNJBKsdHvu3Dn4/X5
EIhFotVps2bIFmzdvRmZm5rxvVgqFYln8vQQIELC0kEgk2Lx5MzZv3gzgT/IRZ8+eRWNjI86ePYvT+0+jqakJIyM
jeOmll/DSSy/Nuj2FQoFrr70Wn/vQQXrj6tWn/nMZ/CRj3xkVdIzVt8eXwCzdaUdPXo05ffvf//7OHbs2CVXf72UEIlEzNV7PiA
RuHTYv38/gPOZgHfffRf9/f1wuyspZOWWCwGo9HIOC1WqxVarRaJRAJvv/4RiZGVl4fXXX0dubi5CoRALuHQ6HYxGI0pKSpCcXl7eiyrgLwUrkiQH
nJ3mtra14/PHH8etf/3tra14/PWpqaliWbvoil8uRkZEBvV6PcDiM3fFx9Pb2orOzE8PDwxgbG0N6PcDiM8fFx9Pb2orOzE8PDwxgbG0N
jYyN8Ph96enoWfX5lZmaisLCQqT9Ph81mY8ePmFeXh5KS0sRjUbR2NiI1tZW9PX1wev1Ijs7G+vWrUNEXh5ruad
uKwokSOWbBF9DoRDjorlcLqa2PR+IxWJ2r9FoNMwrrbi4GIWFhQgGg+ju7saZM2dw9dHQado5NTk7OCKiUSiUTVlzsw2IpIBKJYLPZoFarU3y
8qCuM1PQXCrFYzEr0pIC9UFDrNvAnsUWyulmO+4lCoWATBurIpMkhANY1N9sSj8eh0+lY5i0zMxN6vT7FgJe6MMl
yxWg0QqvVQiQSMTVqCpynUx1oOxaLhQnyJhIJ5r/o9Xqh1WphMBhQlY2L8X4S4lYLIYTJ07gjTfeQHNzc8o5Qff
PWCwGsVjMMozU0bl9+3Y88MADS7o/QlfatK40u92est7nP/95vPzyy3jnnXfu2jnnXeQk5Mz6/YuVcaIPNEOHjyIHTt2MOf
pUCjEHqYjIyMYHR3F1NQUc6WmheTV29vb0dPTs6ibSDq7AJFIxLQ46CE5ODiI/v7+C86k5XI5cnJykJOTA6PRyE
h7XY7OI5jrd7ElaElHo8jGAwyDg+fG8VfKAjLyMiAyWSCxWJhbekk4U+dLHQjos486uQjHR3+K3lf0U2TNFHmg/n
O4sbHx9HT04OxsTGWFTObzSmZGmovp4CLH8jyX/k/B4NB2O12lJeXs8VsNrMOnYGBAQwPD0OpVKKiqgpHR0fR

d3d149dVX8dZbby1aDXnDhg3Ytm0btFotS6uTp9vw8DDzzrrQQ1cul2P9+vXsezeZTDCZTJDL5ZiYmMDY2Bg6Ojr
Q29uLrKwslJWVMcuayclJnD59Gm1tbYjFYuz6IPFC6nwi2xsyYM7NzWUZBIPBALfbjVtuuQXV1dWw2+2zlmrIz3A
6gXxqagqtra3o7+9nVhImk4ltx+fzoa+vL+VY0jnR0dEBsViMdevWobq6GoWFhbBarejq6kJTUxOcTie8Xi8SiQQ
MBgPr0uJfQzqdjumdRSIRJBIJ2Gw2ZGVlobi4GFKpdMZ5GovFmHlzJBKB1+tFKBRCIpFgnVtkE9Pd3Q2tVovCwkK
Ul5dj+/btqK2tZdviOA6Tk5MYGhrCwMAAs+eIxWKw2+2sdM03S6UJ3GzgOC7FBoiOJ/+cIBXp7u5uDA0NQaVSpTW
8Hh0dRVdX17ybTVYCJBIJE26cDRUVFSller/fz+QIaOJFQRz/vm40Gtn9k0qblFnVaDQQi8WsSSMUCjGidSQSSdH
2CgQCTASSLEmys7NRVFQEq9UKl8uF3t5euN1u9Pf3Y2pqChKJhHnxkXGvy+XCrbbbfein//938XMkZLgfvvvvx9vvfV
WWofujo4OlJSU4Itf/CKef/55HD58GIWFhQva/mroSgsGg2htbUVTUxMGBwdZDZjvcUMXCD10jEYjy1bQTYfay9O
Vy2KxGLsIyOOKMh2RSITxi/gPi4WOj2YSq4kYv9o7YZLJJPr7+9m5MzIyPQqUFJSgnfeeeQcbN26EUqm
EwWBgPLILIZFIsI49cgonnngLpCtXW1qZkfJcDHMchGo2y85aP1X4M4M5wNhjOflT7q7u9HS0oLW1lb4/X5236Pgi+9
1Rh6K/EUmk8HtdrNyqdPPphM/nYwEE8aaUSixXzohwaGkIgEADHcZDJCgtLUVpaSlkMhnLQvKKX8fFxDAwMpKpU
KdXV1kMvl0Gg07JpdS7/lvxzDPPCF1pS4kbbgBjz76aMp2/lCdCw6Kw6KVgs0Gg3rdFoMSR
trhNRJpMxjsxyYSXdsMl77v3338fJkyfR1tYGl8sFn8/HPPcole5e5yufDaa6+x7BfHcWzWVFhYYzFzop8Os6p8OSc6AX06
4GCQSCXR0dGBsbAzFxcWw2WwpQQARbfkCh2KxmxwSftJ3DKCyWRCSUnJwgY2C6jkMp9Amrh
8ApYWiUQCgUAACoViXsTrywmpVMrUn2+99dbLvTtzIpFIsMYMhUKBkpKSBjIwMJJPJlKBhbGwMx44dS1HNzszMTJnwBgK
BFO82eh0bG4Pb7YbH42HZ8tnELymrSIKtBoMBer0+5VWn0/7Es0MDAQEqFgUxwi4uLodVqkUwoVQqWXaflry8vEv
6Hc8Y52X99GWCQqGY0Zl233334fHHweAFA+xXbt2XdCwbnR29uLt99+G/X19RgeHkYymUQsFmP+TiRMZjUmzjY
vmV7SzxzHIS8vDyUlJQu66JUQvVHWwUU+O1SbJ54DbhDgII4cOYLR0VHWwUU+O1SbJ54
G/T91VdFsicilxOuZLnRJJYnR0VFWenS73cxElEpy9Pl8U9BkMonJyckZi8/nw9mzZ9Pygi4G4nG4i4G1NlEnBV+yZaI3VR
mSPdzur/J5XJ0dXXhzJkzaGpqYnpF9HnkMeZ2u9Hd3c26qKxWK26eOON2LZt26zqvh6PB83NzWhvb2f6U36
/Hz09PTh+/Dja2tqg1WqZdxLL/+NJ7JLBHUgNkxxCJROB0OvHee+/h1AYeGdyj4YT3kQkdiv6iQkovnIyAg
jX1dVVaGmpgYmk2nGDZsypgaeUF5vV4cPnwYzz//JPe4FsYzzmZsqVD4sxBqLBHu8l3vF55eldWgVtIZm5Qv5Q9xHHLCnVjIiLi9p8hbKhRIjJ7w8+HUjMTW8kmDGr9e
6FVVVVRgcHERHRwc6OjrQ2dTmTzsA9e2DlWzGEc23YHRMx4bEHNSdV7bLWcSf39ku3fIPyDlMNjkyamtQGswhYE4xOP68E5NCxr83p99LYzqfMAXXpsoP91mD/OxYMfL9+MNhBqNqpMSZDS7omt
Xd8Uo0zi+bcwiovQfeSSmvrEgTEYVSVvHpzUp9nzOkHVtyIbmtfYNTsxgKn4hEm9XBjN0W/9Dkzj5NuZmZjzLBXXFvrtmNSeFyOKy7sOzS5SEDX9TNqEYxq99GNU3Bvh9wI2cpeHqrZ9mGKT5Kbz9mhV1y7ZUKHWx3+/hzqTXmjqiq0SXf7z3+0+8QoPlHnd3aMP2DaXLWvQz7V5Gru8ASeYLNX2YpmuFNUdVJq7Rl7BpL8YR+TKfsV+3y+3Y90RpUpEc9EHs2UoGF1QW/mQ5q6/MUNlKiyr3aLiqkqZJlIx/FO9d9a1d2EpmLnCx2z4ug+oBt/5qoGtcOU8y2AB+TNeaR+7qgoeL6ZMvTwqJCp0uZN7Ye9LCgXFRoGGwI2c7JoNKEr7yMxrQgl+3RTdHFMoenM8F0oUcCbkgBQJ3TMx1Y6Uj4ZcK/YnqgZGPgv02CALRt2QRVNx/ksnQrtlQyxvoFKNPT/KkKHH2GVjBdg5TkDKi+gDI2eLmY+ymWjpWLeZCzK1wYC3GL5KmAdHlUa1gKYFLD9jjZHo5ZfZX9iNfL7KaGWGdzKkfUUDHBw4E/ncZ+FGtTA/xw/J7ZAmJZW7cm8CBuXrP6i8ivdv3WBTdHwWs5yeFqTaIEY4GNG2EV3w8rSi3hVMk4WfvA+QnKGU1HFp3Kf8ql6SkSfFi0fA5PQ6Z6q5dCSq6wN2wsdMt0ErnP1LHzSFO5vLGxaZ3CjExCNhPmhE5F+tY1eQWBRdjWM2IFFQi3c4TZWVnBmRjh+mzHhcsaqcmILhC2uJb5vLNqPTkXqoJdJwXbc2rMVePWRMQjEF2nV4ZxgWiFcnuKZe8Rz+9SyIHd8hlyjSVy0phcVvqtmrZ+CJ28J1dMXbfOz4eYrbxPOPxY3zl+pMxHwb2rWnG8f/3HydZ2d4njJvt9jHbp4h0LNgLVdH2Pb9sbOhmHWYw3JHXe+dZM2PoHnGT9evvQWM/+AlPvVLfO9n5h3NGK4Ct5GaSZLY3mBBYj8hb8jnJoO5pgH8RkHBPvbvUKzr7nf/ozQqNN3vdYxwbnhUUk5fZ4zmXOUwnJJ5QE7Ryk+8uPE38ZbsuNbLNPdAhu7WYKUXrvANNb8l0bOyyqRPq5xMbtyo5FbbvHwMmpz6g2uvnYfNrYm6jLjDpa8DvLDYlA3b79qtkBYNK4P6rrlpxzSTXHyGrdRaX6gCt+x+4rQ6AGwx1nXQkOeF1wfV1GUfSzT9fTEzOsWpbDC+0ky+AX/BvXDf//DgLlxRmZ3DXFOIUf8sXe0e1jGd0WMbBhvxTMbeG3TC/FrW0hiJF2YfUHKcaXmR3Km2z0+qRJSrY8qV2JsROkUoeRpL8pzfGKOpTrUbo7Uqd+8NUh2xD5RWgtWOSqMJODa0Da6Zz7wIKSMyGqOnTQWAW6/0vbiXlYS4rS4ccZWcOpQoM7/

OJzweDyM7E5Ha7/czmwSO45hqbnZ2NvR6fUrdnz4nHo/D7XYzoS6v1zvv74H0cojrQTXpSCRy2aw/psNqtaKgoAA
WiwV6vZ7ZlFD3VCAQQDKZhNvtXjaFY3rw8NP+1NVHqrgDAwMpKXK73Q6r1YrMzEyMjIygq6sL0WgUer0eOTk5uOG
GG3Dbbbdh48aNF7QE4b8C56+Tc+fOoaurC2q1ekYHmE6nW9BDndSh6TulrjQyPTaZTDAYDOjt7UVLSwvGxsYQCoU
gl8tRVVWFyspK6PX6Gd17SqVyhrYWneMDAwPwer0IBoM4fvw4nnrqKXR2dqKtrW2G6KVMJmN2J3q9ngXhJBqZmZm
JzZs3o7q6Gk6nk4kikmp4YWEh8vPzodFo2PVNXC2z2Qy32436+nqmJ0UtzZWVlXA4HMzHbHx8HPF4HNnZ2cjOzoZ
cLmet3d3d3fD7/Uyria5zfnfl5z73OQCY4UFGnYVyuZwJP+bl5THuWn5+PutKa25uxsmTJ9HZ2TnneUpdbqOjoyw
gWiqIxWI4HA7s3LkTO3bsQGlpKWw2G86ePcu6akmYkEpi+fn5KCwshMPhmLMjjEQ8iQ9H90a+IjdNZPhcQZFIlMI
PDYfDMBqNyM/PT+m8HRwcRHt7O7vX8ztuSejWZrPBbrdDJpMhHo9jYGCATdaOHz8Oq9WK9evXY9OmTVCpVGzfw+E
wJicnmWXUSpdQmI5095ql2N58sOY4Rr/+9a8hFosZ+ZqI14899hg++clPzvifq666CocOHUq7vUt5fj5y5Ai+973
vLdn2VhqI9EnEzKmpKQQCgYs+4RUKRQphmt96TDflMzi0gaX66YfCJgwBmkK9pycnJQWlpKQoKChhnajbEYjG
WsaIglDgIo6OjGB4ehs/nY8Kb9HAkZVu+JAHtycjkKCgoYP4EyelNTU3C73RCJRObOw
hHAwGmTjeXNBoNDCZTMznizhkGRkZrIvG5/Mxz7LgkwmYw/C+VqZUBZjJXVJXgzowb1YK5e5IBaL5z2JIQI2P8D
gBx7hcHjGd06is0TUvliQXIVSqUx7vS2FZ0z/PIPBAAs4FoqkMwHHdvpx5c+m4jVtJA1D2WxSS5hurClVCplOnd
isRjRaBTj4+OsISCRSLB7MMkY0L1tOohPSkkGagAgDpTZEZFRcWSfTfAh5x8PR9LEJFIhOeeew4x4f/ehH59zepSJ
f9/f3491338XZs2exYcMGdgLSg4ev2QNgRkcYYDYcgskjatvYUPDQ2hu7sbvbvbvb296OnpSZkdkdisXiFKh2WxGVlYWHA4HG
bZhQikQh6vR7Z2dnsb/F4nBdRHZNnsb/F4nBEByWON3xlFn2M2m2G325Ds9oFoRDweHzn
yk1r5Qdf0lmIK8CwWC0pKSlBSUgKLxQKRSIRAIMDEHr1eL0wmE8rKyqDT6eDxeHD27Fk8//zz
Kigp2s6SAcLEPX/KIopKkUqlk3YRE/M/OzkZVVRXsdjs3Gx8fR3gDx8fR3NyMjo4Odg7NNx8vNbDYzCxKVSoXJyU
VqKysRHl5OYqLi1kXF83kx8fH4ff74+ff74YTAY2P/Sdfjee++hs7OTiXsWFxejmVowwSWFxejqKgI0WgUHR0dGBwcCJ1V1LHn8/
gdruh1WqxZcsW1NXVMcPm3t5enDpFzjjUlyRJEIpEwvKMptvvomdO3Kg8lwptvvomdO3e
mteShEnIsFmOKzP39/ejp6UFvby/6+/uhVquRk5OD4uJibN68GevWrYPRaGRZJn6XEf08NTXFrEnsdntKdiMdiKt
CD9zpJTOyTQoGg2hra8ORI0dw6tQpZkat1WpRVVUFq9UKjUbDDSnjUbDSnjEfblQB+Z0TJ+8UMkPOF+apowSH3zdMbJ0mr7
N4uJiSCQShEIhZt5LQYlUKmZdyLzkyQBGW27XC6cPHkSLpdr3mO5VFAoFCguLkZOTg5T8u7v779gJeGee+7Br37
1K6FdfylQUlKCrq6uGe93dHTgwIED+OlF4Pf4re3l5MTEygrKwMP/jBD3DjjTfOe/urwRKED2p9pdbey5lXUs2BB
HwePxwO12Q6PRMFmAeDy+qsdICt8NDQ2spTsWubwDVJKVc+wxM/IYPknSKiIB93bI38WmAIWIrmSuanXm+meMk4X
4XC5W/vV6vXA6nSxZjZjQasWHDBhw+4KQ2sGGAQLnGdmoRsnx5cUJPEGxgo+S2PHJDvM3nsgbw3ssWRYrRu7hkVwsZmdKQ+SWt...pbpGekgVe8AuUe

NZoNKpUopnYbDYRQWFqKurg4OhwMymQwej4fp/5CfGt+RnS8SGI/HWYmCPoMmByMjI9i3bx9qa2uZJtL075K6yKg
rzWazQSaTIRKJoLOzE0ePHkVbWxtyc3NRUlKC0tJSFBUVIR6Po62tDYODgykSEvyJgdFoxMaNG1FdXc0yLVSK8/v
9rJNMrVYzQu/AwABrtMjMzGRaPnybHiqzpDtPk8kkJiYmWGbZ7/envJI6cm9vL7q7uyGVSmEymZCTk4MNGzagsrK
SBbnTg3sSZUwmkzCZTIxzRp5t/AYRhULBOryodZ4WjuOYtyA/w6ZQKNDf349jx47h9OnTGBgYwPDwMORyOXbv3g2
73c5kAYLBIMve8sv9yWRyRlaTfN34GTOa7NFkkiYzU1NTiEQiSCQSrFnBYrEgIyMjZWJFQY5Op4NWq1106Wv6MUw
mkywrOd1zjnznPB4PXC5XShfr5OQkvF4vBgcHmRjuUsNisSAvL49l0viTtNnwZ3/2Z3j44YcFS5ClwmzkawCorq7
Gm2++yd6/66678N///d/4+c9/nnZbq9ESCaToS0FKWlpUu4V0uHtWZDkA6XY4w0s0s5sLcrk8rJ8BWQrwbQXIXJZ
MUtVqNex2OziOu6hOEY7jovFMDU1leJTVXXTGvnzDm9n56+fK+f4q+3oPjCAhUrjp4EYm+L4XbXoP5fHpOrHcNq
q+0zHbPYtF8L081ShUKxaBeSMjIyU72GldXzpdCPjHsKam5q7tF8f86ZKvD12RaKdfX19GBwcTFlGRkZZYRQAC2QzMzO
ZhWN6CZI64aAaLdf0KoGrXqfHo5ESnvRMHO1fcPCL1iAtMKiZiGAB2M1QAkO3Q4zEKEqb8hmiDnSUOeu0nNvWUbB
q+0zHbPYtF8L081ShUKxaBeSMjIyU72Gldt7N5zpdCPjHsKam5qK2RaKdfX19GBwcTFlGRkZYRQAAC2QzMzOh1Wp
ZhWN6CZI64aaL3RIvlgJJPs/NZDLhiiuuYOMSLEGWALORr4HztV3+32iGJ0DAhTA2NobW1lZ0dnZiYGAAY2NjGB8
fZyKFDocDOTk5aGtrY11uGo0GOp0ODodjSUuvlwJqtRrFxcUz3r/YThGRSMRKwZfTlmel8CgECFgpUKlUSz6pnpy
cRGtrK4aHh5kEQk5ODrKzsy8YqAhdaZcAf/zjH9Ha2gqr1QqZTAaDwYC2tja89tprl3vX0N7ejgekdeegktLS3o7e1
lAo9yuZwRIKnrIhwOp9Rip6amIJVKoVQqUVBQgLKyMuTk5Mzrxk/WFDW6Q6xjhq8t4ff70dvby8oqEokERUVFqKy
sZArG/IW60kizhhYifLe3t6O06XriftEpQayCiF+zVwLiVzy+UlqtTrFy8xkMkEikaS
UWiYnJzEyMoLBwUGcPn0a/f39F3WMTSYzzScznMRul1PVspTWO45a
UP3U5QOcCx3FMA+ViEY1GcebMGbS3t6OpqQktLS2IRCJQKBSse8hms8Fms8FsNrPznbojMzIysHHjRpSVlaGvrw9
tbW1skclkqKqQnFxMSPBchyHRCIBuVwOjUYDj8eDM2fOK2tDUNDQxgbG0N+fj4qKyuRm5vLusd6enowdCAq66
6Cjt27IBOp0M8Hsdrr72GF154AW63G1VVVSgrK2OdqMPDw4+zZs7BYLDOsSaxWK7FBYALtmVFKS+ZTCI
zMzOldMcHZSJpP4gXwxweWne1n4qLwRSPFYjEUCgVycnJw3XXXobobKy8kMT9K4GJXatVovNmzdf7t7t1YMNYkx4hv20D
k6+uvvx4NDQ3w+/0pf+/s7/s7Jw1tXmpOEZPP/007rvvvviXbnlqtRmlKbKzs1kHHDj3wafF4PBgeHl7SqJxvZ3ExkEq
l0Ov1SCaT7BisBGJofn4+SkpKJeXB5PJBL1ez4TQBgcHWRBJXJVQKAS/34+cU0NjWjE5OQkRkd
Hmfo58TpKS0thNpuh0+nQ39+Ps2fPYnJyEpmDsLu3btxzXXoLKyEnl5eWm1WMjb6ciRI7j++uvBcRy8Xi+
6urrw7rvvvvoqWlhfEr+MJ4RMikYIJsHQwGA7NjoEC0paUF7e3tGBkZwejoaEr3Et8tbUqU4czzh37hw7N0QiES
qKrBp0yZYrVbm98X3AAa/f396Ozq39/P8bHx9k5ksTxLIuLWI2kSrK0NlZSWkSySSScTieam5t
RX1+flvirUChSGkoSicQ/37mlRlZWFsrKypZ4bEH3ewOYV3l7ykhKVjHXHZ2NnJycfJCNh384Gu6rQ//fbLCoEY
VPt+Jxs3nxAWDQWRkZDCfReC8bB0e9B+kD2GxWJJJjEXJJsamqCx+OBx8VVUZvXLtjtdua
KwHdHIJkImgiupPLibBC60pYI87UEAc57tRQXF+OrXLL788wyuI5PmDwSDi8Tj
kcnnKg4fqtJUIKBhvKBAgJPlUeL2QmIRKJWCYimUymEPaUSiVsNhsMgNrQyXS4lwBFW2THtbktE0ePuFwGH6/n3W
9dyGQJQjNdOdaKPs0nYhIGTe66U0ndvKzdbm5uSgqKrqGT1o6EOGTzFCJQEvKsHQjnU445Xs28U0lL6YzaSGGoEW
r1SIejzP+waWWWF9tzVfmrEXlhsJy0UtRPsxN/ISHFxpbqOmfIYnnOQ8mqR0
VCrm5uQDOa5wNDQ2xLMj0UgnbxLMj0AEQul6clslL2l+/3RW3wSqWSWW1Mx2InN9RJRTICc/0/aXJlZmZCpVKldOpN57HwX8k
ug3//A85n+Pr6+nDu3LkF+wl+WEGyGtMXkYs4hz2pMMi4nDL44XAYsVhdU3PRi6ncx04n4CY3pVms9l
QV1e3pGNes11px44dPw1dw+7du3HddH4PeRJ0UXNTjixeXLS/DpwD6KyVcuhAJzwq8u1jLD4LbcdnJhPUzeMM0b+4i84Mb2jpjc9s0D2vqfBk6yxmB
Wi6Gnpwft7e1YHR1BJBLBZLmb2TQzn5pC3086ZfINNN/NpvNM0o98xkfBRHU1SSRSbIxtNsbCXiUnSlxePxFJ2
VsbExtni9XtYllpOTg7Krq4dQXbRCq4t1GWVlZWH9+zafQc9PT3MkZy8LWeRJVwHHfW9+vUwm83eu3o7OzEW2+9hep8+9Mky8dprecE4d
q27Zt2LBhAwCkZCZpCQCLAtE54rP50vJvJOJOjdaVeKeXEBRd2wzWBnq1jgWwJEQOz5qzZpwcWtWxRcd9dYIeP1ORKNKj5+MGzSOJ
Mh8MBABgZGcFJdJusbdpMsbhc
4kJuby7ImpaWlc5ZtSPPieA4Dr/f3/t8fz3/xla98BYODg2m//PB//3IqJ7gQ9kfv8f94gPhaL4eW
XX4ZEIsHRo0dx6tQpNDY2Xy6Ui9f2kaGOMvJBlEgkSCaTE9p3bp1MBgMrHuR7gvE/1yujthwOIzGxk0dXWhr68
Po6Oj2LdvX0pZj/y6SOv2kZI0aZTJ1Ox3wUQpCdFAgpdRUbUb2MjTp5+8mMDTjxkd1Y2MjTp8+DYVCAYfZPgTjU3oWeY1+uGFg
zW9VqNQoKCqDT6eD3+nGPxaLsSwkq8NOwcVXXSEBoyIx5IzJLq6ILCDNEogIlkinDn2RU9qQ9bgMA4XYPGSTdUVHHUqHo
55BF88YtfxK9+9v+d0+09+Sv09/cjLy/8YLgUn2ZHYqHk5yzpZJldpZy3RyZlBV10ZaJr9nOLTRhdkdqqNVD1cTcb
NasZynx8qlWrWBoN02LJlC7Zs2b3l5eaq8eBwc9wB4PyD2efzoa+RHjvW0tjFH9b8Q7dA+wcbp6KOXcCp9kWm9MwrCBzNaZk4Bhp
Aa665Zs4Z4XxB2lFarfaiAt7e3l40NjZtGBsbY6W40dFR7Y3aoWlBW+88cZFlTltNhu
uuuoqdr7SQ6CmpgYbN6xg9x/SoJqYmGCdX1KpRVwo6OjOHz4MF588UXExO3eAJCbm4uZMmYMmrv+/zvZ4FheXn
5jAx4ZmYmrr/++gXtu1wmbx0kzz/gXt0u1wux0033MYTbbruNvm68++OHLkCDO6Dzy74OjX+vQ2r12
GG2+8JNe0TCbDrl27sGvYCdDrl27sGvXLtaVduONNnnYCCHsNAuvyYLQ3V1NRMUDQQCaG5uxmvvYY//OEPKVk
5ypQTDAYDPv7xj+Nzn/sJpFIpBIiJdUCuYHsNxZP5CuuHcuXNp92uuVnsqD6cDTQzWr1+P8Pv8BAIBuN1
ult0mv8R0QZFUKoVOp4NSqYQ4NSquZUb5d3ccDCfDaDSnaSVu3bhW60uaDYDCI3/72t3j//ffhcrmgVqtx+fJl
/8R9T1iGBv3QPjb/7u79DIpGA0+nExz++kxz66quYmJEhJHyheeh4aGBpYyNpM2Lrq1d
slkTA2gJf84oyQHzEYjHWWbYU9gMxmD97//v//92bmWmCSqNpNUZWjx9kWh6ijCGa
zGaFQCENDQ+jp6UFbWxta29bYxUjM+rB/8BQKBb72tahHV1tWfxg39qtlpl1/Gn+KVSt0kl/EP+u7d
O69evx/r16ITD4cDQHLTwBBTwh4ZGcFJ556CgqqIFcL556CgqFAnfeeSfuvvvujIyMXFMJ88xFuPVd8BSpxOeZSf
+4i/+AuXl5bMSs1ciJicn8c477+Dpp59Oy2lSVVXVa2FrtSVD7DM3N36X
l+4

6aWkp9u/fn3Y70WgU9fX1OHLkCOLxOEpLSzE0NMTkCBobG3HmzBno9XqUl5ejsLAQNpsNCoUCPT096OrqQmdnJ3v
t7OxEX18fotEo08HKzMzEAw88gIceeohlYQEwriSJTlLZdK5jSJntwcFBtLe3w+VyMcHYwsJClJaWMpVtqVQ6q3T
G5cKqKaU98sgj+OlPf4r3338fL730Er74xS+iu7sbIpGIldL2798PsVic1uLjpptuwuuvv85SpR6PB+++6cmiK
XqpT2m9/8Bp/85Cdn/Tu1ThcUFECtVqekp/ly/zU1Ndi5cyd279695KW+i4Ug8JgekUgEzc3NrMNQJpNBr9cz6wJ
SkV1I91hLSwueeeeYZPPvssxgcHMSePXtw66234uabb06rvbvbMQXI7jODg4iJaWFrjdbkxOTkKj0SAWi+FXv/oV86c
Dzuv5EMl3upBgPB6H0+lEOBxmthabN29GZWUl+27JOufgwYP46Ec/Ouf4qmamcOLECbz33nvsHjEyMoLO19e
zbkg+yJqGHySRunVGRgZqa2uxb98+fOQHcGpqCvX19Xj11Vfx4osvoqWlZcY60wnC9EqB9EqB1vr165m
9i8ViYeRnWvj3vmQyiaysLGzatIkbSbbtIkFGrFYDK+99hoMBgNGR0cRCoXgdrvR398PpJaDQKjuNQVlaGbdu2weFwYGp
qCv39/Xj33Xdx6tQp9Pf3z7BlAYCCcnBxs27YN5eXlGBgYQGtrK8bGxhAIBKBWq1FZWYl169vbbM/VIcw1
shr6+Phw4cAA/+tGP0NnZCeB8ZuXOO+/SVMHwdywrrrrvwpS99CfF4HHa7t3
LAiOVSpXyADly5Ag2bNNiASCSCrVu34itf+Qr+/M//Pmm+Xy+tIJ3hEtFvqYHGRHWRCIRotExsbG0NnZuWCpd5l
Mhg0bNmDTpk3M0mN6qtPtdrOZWdtrOZwsDAAHw+H+x2OwoKCmAwGJjv1tjYGJLJJKqqrBu3ToUFBR8aNphlwJjY2MYHh6
Gy+VCMpmEUqlk6q89PT3o7Oy84MymJmn4xNPpJFTidU0NKCvry/tdsRiMaqrq1FSUokMjAzY7XZUV1cvilTOB8d
xTGm6uLj4ogQag8EyOyf7+vrQ2Ng4p1SCTCbdDVdhf379y+La/likEgk0NXVhYGBAdb6rtfrIZVKkUgmI2CRqO
Zcc9ai+jo6MBrr72G+vp61vm3HDAYKitrWXZuDNnzixJF5xarYJF5xarYbFYkFRURGuaeYaVFdXr7ljjlkgk8P777
ICWSLi4tx++23Y/v27at+zGuZOfN3W1oaamhMDg7CZrMBAL7whS/Ckhr+OQ28G5xLxXgbQ9OgEZNM1FA+ye/Ll+NbGKpbpj4XE
jfz2WV2DfexP4Dl72aB0KbyqT6kqJCtzBoUWVVgsXNFzHMlV2UWiqY
//GUMDw/jqaeeAoB5B0YrgXydmJ7u5u9Pb2IhIHqNprRh0hKJRHDy5Eds3d2dSLEkcPnw4rS3KGHc3Nm7cmGI
ySR0mABAIBJjtgcvlgs/nSHC/bMVFqMhmZ+vaBnHk0bGHfsQSe7kuK3SuLU4DxxRuhxMR/Hm3nd91+NDeYJnvy7ixv2rw8POsdhs2WK4bYnDmUhgRHRmz8fyBE66
coOwG2SNMTk6yGfJitEpkMhmuu+463HPU8p1JeNdhxzUTsXcsasSIGGlg3WxYLmQpL7t2n8kTvvN3XsKMr06kF97yJ78N83a2cqlI0cHwJKTZ2c4e8VNX87MPyWTR0vGZ3VxdLtiE7RSwiQZC/X2qCckQfSdG+gJYJOGWVedhdK4RALFuHuR4uW8agbXSN5UwzdOhQu/+jH
Dafexo+qICWVlZ0Gq1zENuz549eOCBB9g9eYSkgZZDaXF5SVI4IyP/NNlNLepz57F6dQncfrr0aZw9e5b5JhkpwPQeRMFN
ZiUQCkUiE1tbWtFk6k8mEqoaDQaZGRkIC8vj9jmYJBIJndDlzbIOnuEjf23duhxbtm1DcXExcnZ1V1z2HvjeY/j
+++/jZz/7GX7rr29729+Y519ZWRluvmlmXHnlltaiqWHGu8sJIWN0AxXz1ql/Ff/zHH7A0ObwWUyMqypQxefuVGl9HXWm
1tbVpHwoSiQTf/OY302aF8JKy5aS9HYLQUEvycx6GpqQnPPPMMp48ib6+PoyD2xXoaRU7ag4qqKcQTa2tr
Q1NTEAhlqG56amsLBgwdx6NChOcmqRLv08HSUlJbjttttY+UOr61tRWNNjY0IBAIzRMMMyTMzUzNkLlUqWPe5xIg4iUtP7iQCHfHYVzCIVotWkv8oqw<
SiwnqZ
Nm5CXl7fsgRbHcWhpacFDDz2E3/72t/NqXSdRzJKSEsjlckxOTkKlUqGqgrV1dXYsWMHSkpK0u47x3FM0JOvo5L
ulfzANmzYgI997GNpya5dXV14+eWX0dPTTA5fLhfr6enR0dKSsU15ejo997GPYvHkzzGYz+vv78Yc//AGvvfbanN1
qpJOyFH5L2dnZrIFg27Ztu666666y66BDhfrFQriaXEahojZbn5xuDpEIvFcOTIEdTX17MSG8dx+PKXv3JeVqXCpf
iGHo8HvzoRz/CD3/4wxTSOHC+LFxQUIDi4mLk5eUxK4+ysjUV1ejsLDwou/HSz3ghTy/V3xgFI/HkZOTg69Tg69+9at
45513oFar8fvf/x4vvPACvvCFCLX/wi6urqsHv3bvzsZZz/Drbfeyev6XVIngTqF5557DtXXV1TAaj
/E9dddx0ef/xxwK3Wee3LagiMlhOTk5N44YUXGHeFbzgAzTm73c7cydvb2vI2+W1dtc7e1EeeHmzJKlJ9ZptVrk5+fDbre
zln5ysyaNIqvVinXr1mHjDhg245pprUwmm0d/fj85cXcebmUp+CiVSpSUKCsrAzl5eXweJh0ws6EHKU7ItHWJO8TTaYA8q9tWPqHOwiyiv4I9//OXQx8Hx2JpnHPtzGxKqrIIZmOGQIBNgPrprTmm7c7cydvb2+H1enrdeeedd/yGcXI+P4EVXI3hD92/1iQ2ZCZkq9hbz51+r0aprrPqqaqNopxh+bgymg7MfuzMh6Ab+YcKp3oI9ZmTmFVcFiHHk9nqpQO+nDV+CWOeI
RIBBAd3c33nrrLWzcuJGJsE8v9Fo9TtTqSXOEVHT3UC3wSDIJmXFD4kT4fTwxJCczCTGZmHGqNjt4qMzt8ZgX9aYFjdvFRHMVbLB3UdMR0P4lcKRRt2bLSndSg5OKZ3pLW6S0CGDRa6i1smQVY5oHCPI90nj86FKMyKE1QKIWvA7TFWG/x9+MaoHm
3Y2JiArm5uRCLxUwxPxwOM4Q0wYo3PkXbhagGkJraysmJZaysmJiZYZ+ZJe33ovm7+LgKvmYmxYJRrLyn6yPbKnqmxGhJSd4jKKm2WAmD3SN8WkbyHHe
UwsTEBPr7+xl/Z6GPy+UwGAyw2+k+r7q6GuDw8sPw+jr68PExETExMTg9jaPa/T1L2R+Ld/oUNrXdAyLoRkCNk5VPcq6SeTZJVY7jeAMDVRvR1aP8LGXEMWmMzTDZfDwY61h5yR2PmV6wKL5HqMeb5/fynMRt9Ytow69B2zxeZ1IqhmYtmsQtGLNAwlTq7dxWrTa5FpQLEM3jJCOVVBM0UlHVVJ1zqkIIBS2f8a6AtW21V0VsbLdbhV
AeGGDg4Noa2ubwcshGU9GQuKcGJGBCHRDTH42OHjh9aGN3KZ2CE8pQLaGEY9EAwPBC5unTmhWM3P4nvgBHMhklcXYWLTlYUtqwPBXXI1YOY1LnOwOqTh3j7SaWtLZyuUUpDhyOa7pp7Rr3J2fgkRn8sVCcrZtRQZ4ZUUgd
mmmsQiUTg8XjQ0dGB9957D62traxkQOYvgEPfJDzjNgDtBg8YGIgpd2UiMj3YcH35mqgaJUbu0zUkSWbgzOIlzmmhFIYlEUb
cTieSySTy8/NRWFiIgoICZGVloaWlhBHm3XC9cl5Oe4FSf5sqp7D3yFbTSu+2uBSxeJ2dWlYKRVmOyYBJdjMGt5EMgTw33uZ3XGHdHRVqBZKWkcm0bLyZ9CN37d8nA2q7Z5Ifz7Qk+rKAxqGQoMcA6fzfNjNJiWrjPmVAoDz+mwp
bmTVIMpmEVCqF2Wxm+0ABH2nKxONxqVq2Gw2XH311airy3HkM5TTRfYeBm0ueRdhMkEoS0o3fU7uQ7mbNGQ9GNG2tdHHz3A6rs6GJxWmFjdqQPAeFDh8cuHLjHQD3nmiJb2nAFiILj0eiu7CFmjXgV3A5y1nD5IcTt0n8I6ap5Fow4BPEQ3H7TqfXG6UjzNt
jR1FfX4/u7m6MjIyguroaV111Fa666iosXLiQTZvE/WGdNnjwgjUEGBA1p9IQWIRR7ZbEhTgUi61Cy8i0U9/jFrJKmDY8vwGkqsIb6ZpQc0mgyDkjtyyrnBMTzQz2J86S9A6A0N5tMIDAk4jJUGk0ri5v3OBMlgLa1rN+mOmtfZ
RlJWVITMzM8UuR6lUnlwmrbg0Ij3rY7QvrVc7ltqKKvqVX9HtHrZ7lmSXCfUsL7zsU15nrGqhB0xyOlWcn6ZfVhTCDbcXZvxOBYsKjuqJdOpIUz9uWvVzeQVYDqgnGxkrpMNhOgCu0sFoyc3KPvtzf7LUM6Chvb79ReV2O6sULqLt3fH1H0D9tR
/fz/C4TBTevb5fJicnEyx7bHZbEgngaGoLH40nRxpqamoJSqYRer0dJSQnuuuOMOXHfddZDL5Ws+MKIu7TvvvP
OCdJyIJFIsG7Lrq4uNkeGRlBW21WbWlpamF7S3r178ZOf/ATtGRO/GRO/zXyU3M6nXA4HMjLy2MNE0JgNAv279+PZDKJ22+/n
/fUZX2rPPPovNmzfjxRxP79++cdGD377Leedd87abgtcOo7RE088gfvvvTWJt+/Mi8cFVJ/8
ZycpjJAamUqkQjUhMiGh4dTbpDw8XRSdvtdqcV4TKbzcjMTzcdqOpgtgm/yF4zh0dXUtKBOMBMB7oB85V7Lxf
EYjHsdjt27NiBiBz33uc9bsitTa+KJRAj9fx9ob29HH0tcH2tvbmcUFNcN6ntTIkpHZNqrUAnPEj93CiAnJ7aTge
j0QiDwYDe3e3t4Zx1GpVMjPzrzfbX9kGl5vMkqz4Kcq+RAJbUlDRJkLH9lVvv6z1zkoHl9HgKOUQi8xUjJnVzJrJL0T7LoXYngHXD2RxaUnxPkp+Q/nKvHYnNo4sbo7JgJDvqV8kKHlGV4DZbW
nBGp8vaRAIICxsTEMDg7RO/tczOfP9/YqFotZF6LL5VpqQRlWtVqOsNbsBBisRjhcBjDw8PPLYkMDAIWFhaipqYFYLIb
L5UJjY+OCm0OWEzqdDRW1tSguLkZ1ZLSwtTt6d9NBgMyMzMzRGnRvVPqTU4OampGqXEJ4lQqGMpa1FTU4OampqoUf8lEIgG/34+IjDVd2kkCHA3ZEqSjiyc1iGrxa8+1MtKpBbS5DIIBFQEA9+u6dI+ETHdXkl3HHp
gMIixsTGMjIygt7cX3d3dGAwOOJvMtRPYgSsQdsbe4oyuGnMhy28tvXk/jGe8a4GUbyfklqsu3hOKJojtKJiSLyGBSDcSQSN5S7mo9C35+XcSxa1ZQmZBF4I8QwMDYH0UalN2YmFWtjZRgUXSLz1HR0/HVFT7+kjv9qB0qXjwgktRBLf6BkqagMiu0ZSp8rpaF6sNbKmL7hBXXtSkvJd8z8M4hvRqRvQDRp6Z4UfLl2VJr92NELdGpMeGVP4xtFbqE61LovUS7tLEohcS8wlaUJRNR3cpdvDbxz3J99hoFFwZ7HBwFPbLBJHQYfa9aoLGw22bKmQg6+nS3hwb+5NP5a3Ty2BD4LtMk4ExcmM2dk1W51BQqLgNXxkWLTsbqtrtYgHPSx47LCo4+Gyiv4tmfTCMHjJdLy0NvDyGfEYU1Ptr2Zq6mgG9m1X2xSFcpLtXFzq8MJ4P8DFqmV9ayJAHI72kbQINNkDTUxNMVAtxfnXe6QGBDSyEfm+LxPFIAy7x67I1WrnSU7o4Dv3hlOaP4K02Mnza2YJRsp4OdhCGFNEh4Ld5eXSaqeMz8kFLIWnd8lDKK/pbpjRhsGl1IRzqWpXn9KgKB7VzHGtr7sLrdfF3I+x9xULvb9ZtFnKmUsh0pdp2FgJPqsULUuvMQyYt0Fzah8ygi8aHEZ1pJrfIYiM5GjA3aLyXGmpsX9UxUXlzTPeR3QsFthD9QyFqtVKJhQmqLlZ7NVG5jiJkjsk2rXjo9HmmjBzcTj0n0mBJNW09r0BAVVV9h1u8m0RFs+WxS9ppb5Or9r3t1e1rRZMGgCwQzK0Tg4ydz8lRh3k6OBKKvDv3jjmNGw0sK+X1ucpZqBdPxB11lWJkyZRXkwxT6NWqX9MIg=

biUSYcCbHcRgfH0dDQwMOHz7MRClNJhN27tzJxCyJg0eBc2ZmJtMJopLXctqCeDwe/Pu//zt+8pOfIBqNQqFQ4G/
+5m+wceNGGI1G9PT0oKWlBS0tLTh37lzaZpGbb74ZTz/9tMAxuhAupiuNcM899+CFF15AOBzG/v378dvf/nbO2vW
l6ko7ceIEfvzjH7MSCnD+IUkn+GygYITUV5cDZHdAWZ3R0dF53+y1Wi17OKvValaKSCaTzBJkamqKdcrR61ygeJ
0ZVrSsJLL5cwclhaJRMICDgoQR0dH0dzcjN7eXgwPDyORSLCSFT1Mi4qKmArzSuvICIVCzGaEvsepqSlkZGQwHyi
6+MkuhrKMJpMJBQUFTIdkZGQEx48fx+nTp+FyuZg6dToQT414H3R+lJaWorCwkNlmBINB5iPn9XrZcqHARyaTIT8
/n3md0U0dOC9iR8vY2BjsdjvKyspgMpkgEongdrvR3NyMrq6uRZc8ydctLy8PhYM/PR0FBARQKRYrY4cjICEZ
GRjA2NgaNRgOTycREG4eHh9HU1IRQKASpVIqcnBzk5uYiJycH0WgUnZ2dzEB1NvE8Ig07HA7odDp0dXWhpaUFXq8
35VqfLfiSSCTMb1AkEsFisSArK4sZ4MpkMkMTDC7If71MT4+Dp/Ph6mpKaaW7PP5Zr3HmEwmrFu3jnXUjo+Ps4C
bH+CTjAL/2lSr1UgmkxCLxaxLj2x7xsfHEQgEIBKJYDKZYDAY2PlH90oKWM6dO4j47xGdjx47Ni984G+j+tFJtb6ZjIYF
3OtC5TveMvLw8VqEwmUzMKHyxcLlc+PnPf46GhoLrpuRkQGj0Qiv1wu/349du3bh7/7u7xbn92ekgdKV9AOpKI7h
cLvj9frS1teEb3/gGrrrqKvzkJz+Z9XNXQldaJBJhnmLRabcp60WgU7e3e3tLAo/d+4choeHmb8W3UjoxmK
xWFBSUoLi4mIUFxcjOkMp2+fF40N/fj2AwCIVtRFjQSjMNpwjVRrGAbeMzPKRjOyR/zKKuwdMhBm7EeKt3HWSbTY
bCgsLZ93/xYYR4ziMjY2hqakJzc3NkEgksNlRXV19YrgXiUSCZwTU3o0MnSVc3W7bZYYXRjyArKwtKpZ
lXEZGRtDc3IyzZ8+iqakJbW1tabs/SWpBksI6ZXlImoNKq3a7HcFP39/ezeRtSEkpI
STE1NweVysWCU9PakUikrPUOhEuh0OhQhZfpFvjsfrJerPUokDHZFQYZQ2wG9hmkhFSofT7fvAI
qkUgEq9XKrKPkcjL/sPhMHw+H6M/RKNRJh1ADOHbDen04m+zw04N3m7aiRJwA
+vj5WFv9//+//4Ze//KVgCTIXHn74ZTjcWRnZ7P4P3ODI6DTCZLmSHk5uaipKRkuKRk1u3QjKmiogImwd+/GP/zDP6Q
1mQVWhiWITCabVw2QwGA7Zu3bro/7/7/Qd0f8qqVswb7UWM7z41JAJpOxgHk6KKO32DHy/8fhcKQo7C4
lZDIZc1Ff7P9frDVPUVHRYlLrttCts2DHy/8fhcKQo7C4
Yx9jPlBELhUJIJBIwGAwwGAwz/B5XEmh8NpsNOTk5uOKKK2Y4QcwGKhWOjY3B4/FgaGiITaYHBgYwPDzMfNgoMzp
fkACn0+nEmTNNnFjU2yg4KliBzIB6P49e//jW+973vYd+fSl/u/322/Gb3/wGN99884K38BAgQIEDA2gF
lxz8skEgkjGdUVlaWdp1kMlkSoT6CTqx5MBmnJbgPLIUXrZ1Fp24M38cMu4mHMYpH0UJyUmMVlOCemIDEJAthHtm3yS5iYx3ssDd3kYyx+rYSMzKKhOGCs
1SOSDH5OSRdqY8RTwT3aqXiDaOhpNW8P80tkDm8xM1LiNoiDTaEzMxCZNFLW
+pYFmxUcHWAYi86Ymd/sUkS+APUd4pAP2pPHp8b8bn79cjR6vyeFXVSlP4D0yUcp4tzs+fMVdYhevUbZVw49rMPAF91BQTSq/
/YOPJyOj6DTtbLbsnFzf3rSZTvPX5vH6Y1cAH0ScnYctKKm+pR03pKqMuxDO9lHKNsfj4FuGgIPy7Kwnhzd0VWlgWWh1OhYc8T8nHbjazDz4WdDfC/Df9ZRQ4EIjmsemYc3/zIMdJx3b4qEKR6dvZiXKDmpO+ELUPQiSrWlzGsGaXUcGgbQH5KFN2bTRMbHmRGHxXuY9+qVwh2ja8HMeJUw+DdMhdSC4TkJEFdrVDwkpkzQZJm+39d+PuZH8qWp61d93wNOzPXCtHt8dSMGhymTrX7cM7SEExIUVJpvvoGkv6RqzZHMT2ZCZZhnz9mSB6Dt23Gd+b9zQ7DqOZZ3Ooz6vD5tXtq9KpsOW9iPR1V9+Kb0+jHQkjNCHm
kQzV2R/LrpOBJvBzTfEt3rQk9mMuHw3JPL/rf/n8AeU7d4IS9cLFhJsUdWeOwFaEzHG0uDpC9DFCa0GWcyGJ94FVUMXgzIvfe/OgBKvVLtCq5XzrXEQGBx7Ch+9QJ8eUIwJ/V8Eq6lMxSBeGx+OgNw2SPAU6+UdYf4xSnWDfrxWOnH8uDd49l6D1q3VLfBYoK4AkYwOfmkCGNtuKO3xkeORbL8CKg1KSRnLUFj9pyCsZt8pD5L6PqxWO5Aa9IpdQ8XZ7LUmAVttJ3OTp/YgHgsJcb7Uv4fV8iUbkUkYhBhJmr//2ELs3BoyoFkeW17X7IPVFEk7ExtlZQv4ePvP8FK7x3T0dl0NPaBWxF1xd8kxVvzrE33MWK8XcU8Vr/Va7Nj61gFNG5jKCjiI0wu6D8/oJ6KKeM31g9wC8DPbFVkCrcWRzTRUKjdrpBDaBfTUb5NLyqDPyZ3+XHmZvmYwA+lpSrbvhGbvKbmMGzEjPB2wHc8cJUjjxxbdkJMoLtwLKIjbWqBGcu98PKfgyM3TGZNuHOsWwB9SLRr7T4yz97Mr3Oe/+q8Z4VFRXLETLIcvqEqf2zyTwU8fNhW+kUEXT3YR/K8X5BoUbzE+z+fjmgiTc1LUoDHlFBdtcXMmQcz4Gwj+fdmL/Jde8+KnInZJ6ZYYVg3HwP7++y1tGvS2tC5BdxUaFyQy9A2XtYvAjXpqTNR6kcDMWbIaqrwPHjBXbmFZCE7mYJ6ccMMHPFbYFaetRlzCVrnBjPf/fmUBtdL+wtNF3vEe+OV8P/vOlQ3LFRySMbZBWGBqOUrk0TaVZJXPsMuLmuvIt/3Z9SvdJWv59hYvAsyGgNqGbnNNy+MP60d/M5WyTfB9PMMnhQ7ZrL18fmlNFIVJ3tH+6T/b7H6/V+ThSXkVrpgOKu1tmKx9sSrTxGVtFhTbjQGQQJ4MTbTdiZbqtfZTfW7F6g/WVYPhTN9i9TdT5J/Ji1wS3Vq17Pmuo8PfiW1MUe6KZVZr7JqK3/LpM9uhNzu/mMjd8SWm6F9VZ8OHm8wvdMbNy4VtDq+M1qK8XDtgWa19Ru0vSVy6x2gj8vLaSJXFEP2fTmtkR1ueeNX/mCphf2JK9b/pWqTBwUnTTZdWmdtI/XYzUkz8/fQ7/GeEsyHmPO/tT0gg6ZLo2vqPNhEfw4pPsEJ3w8dj2fA3w1NIDSoLa8Lx4WtxETyH5c4lxiqAXWcO/fePEFSZ0T5Kx7fTovfmtx8RQ6Rz7Gt7dDp6B2VqY8U6yZ2hlN3LSlTJkFWiVTW7kqA+M7HPbfvhFVNbKJbNSyNAdTyDCC2xP0QfuhPcdvMW3xS0Dw5aPRsWKZO2n5FN5ksGu2x+dJDX/Fnbs/0ekmHo9VNYuHHZvsBAHZUFcqnbjOlFiaHrNzaJZCnrHHoJpgdAaG2pDHjvCvu7FQ08b1TOrLD/cnPp81lTfc9Ymf2nWOCgi5UgJoGZ6v0K+xUqzl/sL8bgWnAg+FHnVLKgi0l/Em0gr7O8zDnPWAPq6BEAQdNrk28f+K0Ggx6TmbWEaWqeDlfyEWN9ZIxnxGpC5J0sgXvzxyk3Y3J2D0dQ8eDcUSxkSJnPz0f4SfTxHYHyUGT70p7oNj8kMlz3hi9Hww3pt5b4WzccoxhsxQJPBiELBjpzkAE3Y8Q+gQNBAA7MPrAFZxyxxaKDGQQt5JSgBwyzQcwpy7ZYGNG4wVYpB8PEb5cztL9c2kyNLFeWkLKLYaVkRHqD/H9DyEVCn3fE3+1ODJxHZOwyrrXzhEJy6AsU0lnZlMN5IpVGJdOQBD4gQDRtmEWiANm0XTxPLUxsQFfvODw1nFYGAZhQkJLQP/w+vYLLp53LzJ8QXW03hDEDGWjt4eyQ4H8LKDbfTJnEkzXK8EBF6jA4Uhyj/P7gDx1H4Os9GfL9Qds+dM4/sfB5AFDCuYkOHGEhj0FtcXWVeTqPewHLxcIgfwfVHKr5hHgFDpp4AFtHVOJMlFhQWVNkVytlePcQ0HsftRjyg/b/yxIZkOCXgNkz0WCqJk5MIkbN1ihkOv0Rg3RqJDNcLYB2m0x1MtKRtzfbP+YtfSLVYGB5HC4GI/7o0LDoZc4MVvrYOf5K2h9K+KF5q5OvfjFI/BGyKfntQ1hE5Em8rcNjRJVtfwZWOcX4zCe6tQdxq9xj/dZl6vSSGnKgpMkEPRZ40N3EPojvJEzCCMdR5XDBe5bSXDt9yzMYR7jc5KgPfSfSbTftNUmMr8M2cqCrgAt/A4TtzGP7wD6+d2l8AR6G0wb7LhAM+gvq1z2h/CuQ7J3z3m0ZjDDNLC0CYFdnTuEF4F0xrY01LlL5znWHHjcd3AQ/l0WMhFUV/DHaHr2mjkNq0NkezROPgjkKzF5SDJLYtN2zMDLeWp3Ug02n4Q6l0HGl14bMWiA6tHNUILmIm4XWDiZBjHM18/MkFtmAy8xl4ZKAs2VEnM5XwdBYfYc+4kWsyXd22SBBj1HlHEX7N1Hnv5Y6V+Q5e4q5vMPQ1BAMM3aTXu21K0ER1dVzMmrj9ICgTl/2jp5Ud8VnYoxHT3MGxUgBzzXkScvz3lS26dECX4UD3/jKBoBsOiw0xdIRbshuHtZGF5Z9hKH2ZlW8SkNBYn99vWT4/P5AQBH5vKhYdTdTUBMlbJLEZCsNHBWbv/zpjWNL/SrqoL8dT3s1QQvVrLknIj5m6PpxQIE/NmKaIEUs0zsF2uA6wDSJp4HLp+PZRs5bhUYLQ64Jm9Bb1XUq4HmB4BAARyqcwXYGdQrFgZpdEEC8YG0XU5eF7awEl+qFxFsH/5HtQy+XBdSkQ9ExSpf9aSnD/l28EM/YFDg3iaG5i5C+CKuH5u+72eXC48m8V7LpU54d2BHVgWpIlsQ/7tmAvGQoXTOtAA1MxqnpEuSebrxYJE+BL3EOJL8XkwkyX/k4VzvSKOmp4rTSPSPtKDYjELQ2jiJSWVlE7Pjn4MyWfJcHLSXaoYFxlX2Ik/C4KBhwpMNuhg0rrE49mHDm5MZmAz7mXoO9aZFNpGnUAizW0v4CNYMJ7+iXOp1EmKpvp5hrvq/5FVmUYs9kXuvN6CoHwkjlfaU7GlmwmRkKqYXQ78rBdX57bvE5P9v2Qk8HTIY1fwavBUyK0cj03fmT1W7xDqJECAZsTyJ07P8jR4d6/x1H0K2jsgwYMBCmnbmgPzNTDjeqLbTiZJ0SVHUeSfe7hRJn1oWpMwCRn1C4AUKVWHf5ufNcGQjEgpx7LR3ssVrm0zxNnUkqBePvRqlzBLlGRAFMlSH5RzOfBUjdiqLWfNT83CuBYVEoTBbxjHBlgqL2nSCxD4s3eg9aQMuTi+i3LStaJ4c8RP4sZOeR9DkAJVRt18r6bzmhx/pMj6BqMPS47BHwCgSfW8SlP2uHs/f+ftSQRwm8f9A7D2DC3Myea39iUw9Iq5P/UL9kT4kdtk9pKahHY/3ZJdUgQy9Tzm2O7I8tEIPj+HWaPBLCjadXaAXxGTzbE4owVWaSg0sLPsaiFH/N0DtGqPxHJvkxVhS8gK+VMwXj1r8VZC1m3HBtPsYB7UDVY5E/0tr4lrsuzMYqqHdvqZA2tHMhk9+ZnU+Rw9W4ZT0QZ8LBTQVJE3qH3UrDgZ7p3ZgUEkuwB+ZWaUT9YmtdqXPmMvjuMZ7kjvRsX5Zn+DbNn0JDD1lHtlm8OYXwA2RxWsyEg0JtrmAv7IQX7tmJ+NzCGMAyOA7IA2iDGWBTyyI7AERsSm4JpKBWfmLxoaYkMHGTx3fzWIPdN/FnwyxD/XcfQ4j9HDSsk1S3qV82KWbZrT3rXdKlzbmcZyNUU7xBrBn8OLSjYPQ1r5QkADLUPQzoPSHnNAgt7PU/xUmQbJ2tc9zNVJBwXd6ZeVm1jQ1/5WpF6y6L7Me4/o71lA8EZHjNXW0RZtkTC4nK79zcwZHLkuqXPwuM7G9rwtbs9ey6/d+wd6aiRPb90/QvjvgeZl0/WtU5/tz8eQPzMJXqdMlMR6z7jqPJC6X2O8WxVUJZ/f4Jno+x+tm/Kx9K/5b8EBDY9P4qRmHl1rIVPT2Xm/I1cNJWPvGrf8RAlNgGS4O+pm7dTRvlLDcq5vEZ/P6ejRNrRxjGJSgyCBXqUbTORlDJXnr0/JCKcdAy6U74HOFWoTMKdjTM9OUOWc6nJqJYWM6bDgPJYjixAuQPwcAM7rTpbzjOlSgvdGCbtjOcYbqfoUVO2/4IMfEOdAsOV+IZAwDsQTx6RjjPkpsZ5tWgwjkq9hHOfHXB+4QCFexH8/M3yHsYwCdWdKOuqg2ZjDPrJ8CpLf8nJrHqM4R6eB0PGrgJbYPvMDMhmBNrnaAGXCMGZ6zvdP4vo/TaIcYdfOjjDr4/zrdzmqT8t4M73Lp/ZbLtH4gWJCUlJTiwhnV1T3H1Bnd8s4Pt0y4GuN/q6TlaVHzlkiNDwXRtEJF7hbdwntm4n8eFXFW3yYsxZ1yX4OKYm5HjfzF0MGbLRdl8YSxngYC9yHlxQWZE9twg/9G06LqsJ5V9d5D5L13X2rzfd+c5X1BuNc4yrltztH+Mt8jRK/+/Xd5kOMdkTrRhfIFNRKe9XCM0jyGIw1QHPu3/oOQK2yn0G8OkK3fkNRE59lCxknaBp7tnKYGvRnMXZuhADz8b5Pp8DAGknCrNRj0jKeY6g6rXvqzy/6Mm9EN8ABwsZIC43LwqLT1JnGlb8qCh/HtJoQF8F7t3vfMUJyEEyDMfyODG9PKWM2UbkhOOfpE78W49A8CHDgVkRf8TPVPmKW4nHrPdYcXmFJzT9NMqtNTkYwFXCqwP0wRhGN8rCDP7FNkZH1sU8Xq+bsY84JhDSTl0ZZ7lKL2W4S3A03ZFgqI1H/C3pCXeHy3fkZC2Ik7iTzTQUmwtIZiC6dt+6g76nZXxXlMhZ6wHnCJs9WFSgwHQ7JfOZvMWpovsYPvuzjz3L5DcMCo+NhP8jbYHOtv/A/1CjGKpOkx0mUYmC1o5jNDGA5aeB7Hxo8FpcC4hfjrtXy9Z6TpvSMT/lQnD9lJsYwGqpV8j3snc79Qr8u3e4DCSyb6CSGyGHYOWbuI4D3/Ty0gE5QymMPnWt3dWAxbRcmPmNF7LkdKqFDvPmBAyZhHUO2TfIt2lLyXIG6dgqwDr+mI0kQO1ycmV4EtVvMCU7GL3ZW1KdbqnxlRZ0PT/Xg3Pr+t5WQIN3OKBZPI5rkHG2m6QXgy4Up18rzbE4szcg4+8x9ZIZnuRkIRGrsS5Px5DtKcmKlNS+D7IOKcCIb0fuFGB+8h9ZgvaZM7VqSj3Qu01I8aRmkDuFbKbr+5bGpLkbMRYvEKnTaePQbE2/pVvPgKeMVgkxrLY9JkgmWR9bmFbT4jPHvHmmNZa8i+UX6TzbE4EJVEPEZN/gPK0HlkMZG0cytKjDa6tVjyCCBcnHPNQvLBOOt9THvpZPgEoQcCSYhKFqrNPO7n8IRcrn8fNsW9DRCPRsY9YRcPGP4KygdNY9iy2SlxIaAIrdpYkUWMvE6LKsIw2mzypD30oPntYWJ4jPa/sV4DVpr63cnWxyhoZlxJd/tbFy9PwXY4TIDAiN2Ef6n5wT/Y7FjxIkfNyldgMLNS0f9ZPUhN6VDTKSkOFTNNu2mK7XNNwg3R/Tg1WqU17O/c2FcJT9LMLi+AtuQuwhoeMIE6AZZCuqjgB/Kpc7bgVhFhXVrEu3/Rp5LFYtc5VbdNLNEm+Hz5ajy1Gc9Al/5Nm6+LEyOh4U/ZPV9UGR6nOnvTDPb+efrm/iVWSgWOYo4VHfXWwv5nQOGs3yRJVIN5ikwqxj0jBt/Kxn/yB8MRL69/G7vRh1+9qN3mF2Dw/gaWvaw3oOQp2xG8ZQoGk12c8p1z0q5TChd7BpV+XHt1kk2OmDcK9mXWmkY/9mvXeG8vYz/2bKBUwV2LsM8nKQ8M1jrOdy9t+nKN8XmR7h1GqrrmGvG/8HxfkwEjRgZNKA8j7RTR1Kjr7qQmwdqPiN9WFkz8x4K6Mmw1DntK1tBVlrpjQ7/pHZD4NSUH5EqHKcMw/6qz+xmIufq+vDQGhcwA/TqjqgxvMhqbrUEZ2/hldSvq2mDrQ4Rb8jYV/qKp/rmnRyKvpqpqp7Sk0qg6n58/BCxnsOHePJK59GFVQJXDhfRJYxUHr9+SqLqUxCRqKW4ltCqjoLLJMmbTHHw2rIlrImXpr+vcrYHtpHd1o0Y+Kbwsb7MPMTLgqlEQ/ZHAVEhz1qltmmjWKkpYAqOdXOVt2a8eK+CvYi/BqqUnZvkn9R0T/KXZcaQJRHnWzwXTSHrbWP1GKzKYGy3tTy6cgGv9xTtX6RpuCB9YC2j+NmGrxOh3pnTqtupKkiT7L5qqDKQaaBBvW3Svv/EAYV8ZKu5dB5sLCQ7Ryw4E+r/ZPbF3nbV+lMO/ofd4rMOt9VKSvaWS7vTAsGjTPcSFnRAjFtD4vOoM/sfF3gNx9Zw7xlLXSKqVr4/D+JaGQ/r7CF5e7L/o3qaeQRy1JeGYz3DgcHRsLoqaJYfjR1sJf4d3oOWUb3U7jRyTDiqRlP0PDd1UrUKFKDeOOt6Qr24qqlRBO0PL2NUnFl8YJeHTCaMkQ5dTE4CWJBOsqz/g0TH6tVo9d5Mt9pa/EuK+dUbmrV3YFL7fU8lZWPuQbExHnN4dZc/OyEWpmNn7TwfjLnN7rLaNN5Gs9lgf13dsrlWt2ITJcLWfqyVmAOLxP3f6WbLNtT9NORvSOQQAXKvDBqb5ykbZHsKIjWHQCTB/mWSdFwDUJwyPmvnZdOQ5nBCMmcPOUJrCJ+FSb3E1sUjIfBAz+dB5MWs1gtKeHXpR+Pnp/7xdQ1fy7cSE5HYZq3jdmCBMg6MlEXdRLRV+KiFy9M1BZs+lxHEJQ/dM1Mz1zCuj5GY1h2UsQzq9m63KPwmq8h62x/3r3cKPlAeMYeLOnE3QA3GdF3Ysnd/T6xhz6oY1KMGNWV2T8h1mhONCn8XtJZZkMtiG2Dlu4Jh1pPzlB79Y5/r1AQvxo9aKfRxNqvBUYgM4VS89WAvLUDvYq9LNL4hD9fH62W1X+tPn1aa5yQy4YDL5ZjFWmYnwm0X8Gv6lgX2HAuzX1LAY3bpMOhY9cXMEDvH6/+sIv0GYeh3zn7oQbw5v2z3SmRoCl23XXpoPLyGTB7YiNsAiZw2o6cmfgEBcOd5qwJ08gHnF0GX4qS5Q26aZgOSTEJEvgQqoP4/OnaxLPtXWPlc0YZIbTmY2S5pEefgUsYw1v8b2VxpPMV9KTo9XhlAZvO6YmJ/t62zL87Dv3bz2zTsUXfdS3o+Vl6Vsf1cDA4ev6vjUk2aGP7YlSS+nEYWMs0JmWTm9Ba0XqlBUDdDm3/gmg0RLJNYGpNr1DNlZoFNwuu2k8XRCnJBJDhLXhO9C4VBbqgHN69pMLrm2Qq8d7zsOgPdTUlK0SKe0bIEvDGWU3k0N7jlPv99vO6j/Zpn9Y6R3jkzbnNkQ3lEf6FjZDeLhDX5+DKK06DrMH0n1+7L0eN3sj4eVRfKw49llPyGM93AOHdj8HS7KxZAMNbC+KSIq2nqAy9eMD4Dc6HVqs+0wImwOM80i2DwrXRGSZ/Lk7b4b1rSxR55MXFpA==

eDFbDdbiUWMnX4mKx2q7DL37xi/jDH/6AgwcPIicnZ851L/e1KARGSwy5XI5NmzbhwIEDKe8fOHAAV1xxRdr/2bF
jx4z133jjDWzevBkymWzOdWbb5nJiMWMEzs9Q77//fjzxxBP4yEc+csHP4TgOp0+fht1uv+h9XggWO77paGhoSNn
3tXAMgfNdJp2dnfj0pz99wc+5XMdwMVht1+FisVquw6XESr4WF4vVch1yHIcvfOEL+P3vf4+33noLhYWFF/yfy34
tXjR9W8AMPPXUU5xMJuMefvhh7ty5c9yDDz7IaTQa1jXw9a9/nbvvvvvvY+t3d3Zxareb+5m/+hjt37hz38MMPczK
ZjHv22WfZOkePHuUkEgn33e9+l2tpaeG++93vclKplHvvvfcu+fg4buFjfOKJJzipVMr9+Mc/5pxOJ1v8fj9b51v
f+hb32muvcV1dXVxDQwP3yU9+kpNKpdzx48dX/Pj+67/+i3vuuee49vZ2rqmpifv617/OAeB+97vfsXVW+zEk/Pm
f/zm3bdu2tNtcScdwYmKCa2ho4BoaGjgA3Pe//32uoaGB6+vr4zhubVyHCx3jarsOOW7hY1xt1+JCx0dYLdfh5z7
3Oc5gMHCHDh1KOedCoRBbZ6Vdi0JgtEz48Y9/zOXn53NyuZzbuHFjSmviJz7xCe6qq65KWf/QoUPchg0bOLlczhU
UFHA//elPZ2zzmWee4crLyzmZTMZVVFSkXOiXAwsZ41VXXcUBmLF84hOfYOs8+OCDXF5eHieXyzmLxcLt27ePO3b
s2CUcUSoWMr6HHnqIKy4u5pRKJZeZmcnt2rWLe/nll2dsczUfQ47jOL/fz6lUKu4Xv/hF2u2tpGNIbduznXNr4Tp
c6BhX43W40DGutmtxMefparoO040NAPfoo4+ydVbatSj6YMcFCBAgQIAAAQI+9BA4RgIECBAgQIAAAR9ACIwECBA
gQIAAAQI+gBAYCRAgQIAAAQIEfAAhMBIgQIAAAQIECPgAQmAkQIAAAQIECBDwAYTASIAAAQIECBAg4AMIgZEAAQI
ECBAgQMAHEAIjAQIECBAgQICADyAERgIECFjV+Na3voW6urrL9vn/8A//gM985jPzWvfv/u7v8Nd//dfLvEcCBAi
4GAjK1wIECFixEIlEc/79E5/4BP7nf/4HkUiEuW5fSoyMjKC0tBSNjY0oKCi44Pqjo6MoLi5GY2PjvMw0BQgQcOk
hBEYCBAhYsXC5XOznp59+Gv/4j/+ItrY29p5KpYLBYLgcuwYA+M53voO3334br7/++rz/bbb0dJSQkeeuihZdw
zAQIELBZCKU2AAAErFllZWWwxGAwQiUQz3pteSrv//vvx0Y9+FN/5zndgs9mwZGBf/7nf0Y8HsdXvvIVGI1G1G5OT
k4JFHHkn5rKGhIdx9993IzMyEyWTCrbfeit7e3jt7e3jn376nnnsItt9yS8t6zzz6L2tpaqFQqmEwm7N27F8PgkP39ll
uwZNPPnnR340AAQKWB0JgJECAgDWHt956C8PDwzh8+DC//C+//v41vf+hZuvvlmZGZm4vjx43jvjx43jggQfwwAMPY
QCoWwZ88eaLVaHD58GFH0d/fj0OHDjX9jL6+PnAcB4fDwd5zOp2Ix+lO1NTU+jq6kr7GeFwGACgVCrZe+vXr8e1116
WHGQyWcrvIpEo7XvJZBIAkEwmsWnTJvzmN7+ZsS2LxZL2M8xmM4DzJTVaRyKR4MCBAzh27Bi27dixAzfddBPuvvtu
lJeXY8eOHbjzzjvxyCOPYPv27Xj00UfxyCOPYPv27Xj00UfxyCOPYPv27Xj00UfxyCOPYPv27Xj00UfxyCOPYPv27
/jIaGBsjlcjz33HPs701NTZDJZKiurl7WMQkQIGBxEAIjAQIEfOhx7733wmw249249Zbb8WRI0fQ09ODt99+G1/
wODiY9n/EYjH27t2EYjH27t2Ld955Z3fOxcC738HJkyfH/Yh713/PhxfOc738HJkyfR39+P3//+9/jyyy/x+9/
wOA8DCIGRAAECBAgQIEDABxACIwECBAgQIEDABBAgA8gBEYCBAgQIECAAAAEf4P8HTngHO7h9GJ8AAAAASUVORK5CYII=",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plot_eeg(trials[:, :, 0], 1000)\n",
"print(trials[:, :, 0].shape)"
]
},
{
"cell_type": "code",
"execution_count": 16,
"metadata": {},
"outputs": [],
"source": [
"from matplotlib import mlab\n",
"\n",
"def psd(trials):\n",

```
"    '''\n",
"    Calculates for each trial the Power Spectral Density (PSD).\n",
"    \n",
"    Parameters\n",
"    ----------\n",
"    trials : 3d-array (channels x samples x trials)\n",
"        The EEG signal\n",
"    \n",
"    Returns\n",
"    -------\n",
"    trial_PSD : 3d-array (channels x PSD x trials)\n",
"        the PSD for each trial.  \n",
"    freqs : list of floats\n",
"        Yhe frequencies for which the PSD was computed (useful for plotting later)\n",
"    '''\n",
"    \n",
"    ntrials = trials.shape[2]\n",
"    trials_PSD = np.zeros((nchannels, 101, ntrials))\n",
"\n",
"    # Iterate over trials and channels\n",
"    for trial in range(ntrials):\n",
"        for ch in range(nchannels):\n",
"            # Calculate the PSD\n",
"                        (PSD,  freqs)  =  mlab.psd(trials[ch,:,trial],  NFFT=int(200),
Fs=sample_rate)\n",
"            trials_PSD[ch, :, trial] = PSD.ravel()\n",
"                \n",
"    return trials_PSD, freqs"
]
},
{
"cell_type": "code",
"execution_count": 17,
"metadata": {},
"outputs": [],
"source": [
"# Apply the function\n",
"psd_r, freqs = psd(trialscl1)\n",
"psd_f, freqs = psd(trialscl2)\n",
"trials_PSD = {1: psd_r, 2: psd_f}"
]
},
{
"cell_type": "code",
"execution_count": 18,
"metadata": {},
"outputs": [],
"source": [
"import matplotlib.pyplot as plt\n",
"\n",
"def plot_psd(trials_PSD, freqs, chan_ind, chan_lab=None, maxy=None):\n",
"    '''\n",
"    Plots PSD data calculated with psd().\n",
"    \n",
```

```
"    Parameters\n",
"    ----------\n",
"    trials : 3d-array\n",
"        The PSD data, as returned by psd()\n",
"    freqs : list of floats\n",
"        The frequencies for which the PSD is defined, as returned by psd() \n",
"    chan_ind : list of integers\n",
"        The indices of the channels to plot\n",
"    chan_lab : list of strings\n",
"        (optional) List of names for each channel\n",
"    maxy : float\n",
"        (optional) Limit the y-axis to this value\n",
"    '''\n",
"    plt.figure(figsize=(12,5))\n",
"    \n",
"    nchans = len(chan_ind)\n",
"    \n",
"    # Maximum of 3 plots per row\n",
"    nrows = int(np.ceil(nchans / 3))\n",
"    ncols = min(3, nchans)\n",
"    \n",
"    # Enumerate over the channels\n",
"    for i,ch in enumerate(chan_ind):\n",
"        # Figure out which subplot to draw to\n",
"        plt.subplot(nrows,ncols,i+1)\n",
"    \n",
"        # Plot the PSD for each class\n",
"        for cl in trials_PSD.keys():\n",
"            plt.plot(freqs, np.mean(trials_PSD[cl][ch,:,:], axis=1), label=cl)\n",
"    \n",
"        # All plot decoration below...\n",
"        \n",
"        plt.xlim(1,30)\n",
"        \n",
"        if maxy != None:\n",
"            plt.ylim(0,maxy)\n",
"    \n",
"        plt.grid()\n",
"    \n",
"        plt.xlabel('Frequency (Hz)')\n",
"        \n",
"        if chan_lab == None:\n",
"            plt.title('Channel %d' % (ch+1))\n",
"        else:\n",
"            plt.title(chan_lab[i])\n",
"\n",
"        plt.legend()\n",
"        \n",
"    plt.tight_layout()"
]
},
{
"cell_type": "code",
"execution_count": 19,
```

"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABKUAAAHqCAYAAADVi/1VAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAADR/0lEQVR
4nOzdeXxcdb3/8deZNZPJ2qRNmu4tpdC9FCgUhAJtkVVFREUEK8oipfL5uV6vRZFEH6X5QqiXC8X8CKCGwqI2CJ
QLKVQWrpDWbq3SdNmX2c9vz/OmUmmWZplkplk3s/Hg0cmZ87MfL9tyLfncz6fz9cwTdNERERERERERERkEDlSPQA
REREREREEck8CkqJiIiIiIiIiMigU1BKRERERERERERQGnYJSIiIiIiIiIiIy6BSUEhERERERERGRQaeglIiIi
IiIiIDDoFpURERERERERERZNApKCUiIiIiIiIoNOQSkRERERERERERl0CkqJ9MDjjz+OYRjs2rWr16995pnl
jBj6fD8Mw2LBhAw8//SA//+8scpIJD14svvsiyZctSPQRlRRiFm8V5cUBYNaTWK8cidiVFWdH8H9aAtdVDMJBG
AOnToEF/+8peZMmUKL730Em+++nR3weyHL/qQZ+uK+ErfNbNjYiD3Uw1rDYNBtTehZE3HOdy4Or5+eyf9mMp0d
T0+ii7T/hnYDH+X3Ze8hgTYgARM9exTo5UjP5zYiZSf1kFpAHdbTDG6Tq8cF7rHGasFpW2CNqvKCCkf/f7tU4t
c4c+cHDCgWZ4H3Ld2d2r/Uma6Fr/Te6/OxZcs3WqDM4Pt/hhDFO4nqMRQFMQjPBHOxyB6jjnVDdPuMIT7uR2cHoVo
HQRq8rrGnVKQPVuAhA7w/F4nQN6v+2tGXGGqxNsfodHoCyu9BncyZr7iipHrEDeH8uN3TFq7o/B5CXFpo/9bBQyYv
5KR8qdiSvRQhbRjcCPaafnpM3gRjWWRwCP6IhvOB6vp9q6Odbi1xDN3XbgHMnaOZvWX47Y8EfEXYyT44FtEBx0v/
ctTQqKB7uXFHPc6sORV8uJwCJnL4W3+umNNR1XEcw5c9i1AiVhPqnHK/O2Ng0f4uxQqkdvvpW5B6zIgvn5/hb3DLQ
fEuCIRL7NHNTXz1I6GfYpAHE7d9Uu3ZOsmuDsT1nx0b1/0l4o7MPFGPPzk8lVjbRa3sDs0MhI1fbJ6pt7hPw1Ia2
SemGqDVOx1GI2Pl+WD+JDhr5VwMobcgUHwHGKoPkhMdRrEfJwb1Gw7Ik1x2DnzP1BYMdxs6QJb1VawFEg6OKiW3EJ
xo0TPPO0GxcWPA6HRYvZ64Jtj5fEdbc1cxx+wEf6ihRHFHDfSCBp93+h0F/ffr3qlEN+G6I8U+gCnkm5LAqcD2NU
Elkis4ApChNm5z5S/9oEKymCH0X0mPMZZtUQnM3xX4JXG5Jgi8hFDepWSDVXXY9J8xr1XI8fvzecC6Uh6tgFXzPsA
3xNIDsTYFZvvtXEQoOMmi3CtN0jUJ0Lwr/1Rn7CHZ/ER80r/oyD5K1TSVzuZDhY8lWLN9Z6YW/EngsqbThEy5CKJ
kZFxvcumiHiWKDpJcFB55wbNp7pbOaoQUqINfL1jiubBL6ogSg1wW8WR2/S2oOCkD7tR91OsDYHTu4bmYSCMRJhI
ylzIeRiYYSGe1vBJAKgmAonJ5IRMOIp2s7n6ViObkfyh+gj7YJO0yO2WkOrCuYK6aFOJn8vFDFHlQtfqBBdNHFcLn
1IjFOCHCMeo3p8jnRhlOE0ERJ8Of63V+aT/ekHCIQQ3W4/F+E99choiRYYOdtjSc3CnTqzddsgJxUdOh4gAnpF0lG
hFwVH/03BZthYyBlzQ3J8KE1OYuN/T+b5HoSQgBgnQ/89YqZrWIz2bzTJ4UaLJ8QNUfJ/H4jaHq1N6SJoEGn2oH+
AjW6BLlx9cT5+LSk8srNvnMiKDy2iKc6hFV+AUHR2H4AzqQ/SwWmQ3hKtTCPwaF24C+OdIRRjdyBpgVOzbP/jbd
rAt37pTTHr7H8qtlw79kHyxjQIbHHNOn/gNbaTx2Xm2mtbW9tyAUyJCF/T0ovQ2fWPrQQs9D/69jv3PrwAjQ7AqO
QIU1bBmXcNPqctZy/W+c5HHbB7pq9o/sNzrQf3XfJe1yKvdLHqoHh2Sxy3mH2gMFN3hNHyoHPHc8jp3fG98S74tJ4
83z4+gTf9bpuZ8HMC8nT+a9V2/ya/Ppp87wBX5fGPq6Ht6cBSn8z+nH7V5vDjPaMUWEDuP+PKg0xLevjSnZijkMo
wu/QyIxrNVvjqVUoaRK5GfEPNFq6snuGN2aUWXO+hbTDhT0tsNfUXxDIhNRbTQJCfn4yqyiXB3XfdtaaK08+KKSeV
S+9wgwctlCrxORZq3ntSDIvsvqq6cYh7SwezcLPCvIe2jqdNFt2KrnscWWGh+gRAjwm6XFw31ffFKJPCxgpp3NLn
UrEeILfYVSrNrbHngdf3vbnsnfibD8JJ/mLZZOeYK/OMmVgr1nYt+5hL5yrIwXNJ1NkpYvwvlsCJEVLi6CyPMt0K
68bTejutNv5TQwDrCAQ6d8bPEQrbTQu6Tyk9/IKmQ7Oajvmf+gGNuUmfq8bTvbxtVRrPF1jKj2J7cFWd6iSOT/Kg5
mg+OXm+bRGYPBEPKL9a3mY2BEXH8OVAiQwTcjgHFlGm+d/ne3pntxK5vf9AUUrk3WdGgUgK2mpa4Gzb09EnLrT0Z
mA8sCVixpx1BMA7rRBSPMbWVWl35fsP+BGIHtcR/PjMKJzNFGQwjvrSo0T/x+zwK8dmhmV0TntyLfqe6/mBWdDZ
9AZiH5Y3NvLKk+jM5tbw5yn8SD6yo2pIFyajaHcUKUe3Ztaeh+3RUJ5tmuUUk1kTHAC+2NnRmaBTdSx9QNLIs1/D
6QmBnaTn75ShUFLO2ZqF3ciMbLLQMN5ZH3Hjz8XqiZuu8j2Gn69qlDzXvu5TFJqJGJBJvp9d9kptW/9zfpcaaT+ys
hqOPv52s8ttRhjkeadlvf0z17mS+f/sMUk5SYlJLRNZsVOJyuQtnNtLbqbxEVgjzxoUMSfmjKTKKmzd/61AcAE/lL
ff4Xf6Sd3LBUN6NYW/o1xL9EHpI55gFx1u0L9Cp3i2M7dNmNQ0f6+/rzBK+vK2aaRzTxm88yixTo7nTPtaMp/xrD
JN4VI4KOXZ++8v4r/bXyjIfTbp5o06qfXZc8rNZb5xGYO2gr59dKiVr4DUNwa9jlAtaF66G8d5T3A+1C96s5f8Hu
+NR8/T5Xg/yvz6XfYxv+p7dnCzq7q0Ro5mGHvYTwqJznqbR8wE7hpOJzgk7oFYGpHF0yIopkWr4bd/lTuH1xLPqY
aktNdr+x7uNaPE9hAhRe79Wr+s7Ia4ev9GZyZTxiPXNaH5YM8sdwOP9/kf1X45kYFgbI7lI+a4MPeuhiNo0UNwvXv
xa+jvI8RdjnngnBKtFkc/QHpTd6JIj+08Hso4Ua5sNYEAf0JA1rBuZjUNB4uTMhyk8ffoxu+LtgHN2KnSe5C73Ge5N
wbhyeezNewwqMG6TN0oYo8XvsPHwNQ8DAvwxc0D1r3jZMH1N2xiiDhJ2jvvP7DLAYEb8zfDhRUU/bgnoqQALVD33
mh7dwP2xw+2qKTyh01VsHefvRXT1BZD/Y0RHEuJC8oJLHMg7Q/p1/RAD4Yk6Gcse3Ui0ul3GJPYeexIx2YO8qyna
2Gnt+xG5rF1pyh9uDEDt5iXeGhgdHBJz4L13R24D5eLhZGUUfj1euV6MnKoFwNHR0zfjAS1fyW0gd0F9XwYtkfgeCr
xXpcxi+ZRUO6ZqoEqOYHpbgVRJQQiZJLa6F9U8BzzsTzuf2UBhKqGrjxEzGUuFUC4p/lsn9W+WRvFpmzpXDPk9TG2
cpiPYSqf6IWGOZVd3jLX9/Gk4C6XkrHkpiAFFLzejM9ulLXGhCRUPLdZEthS8nlnYDg3QZ3U6OyxIK1xlClmXu7W
+0+MQoRrD+iU5TM25NySz5xF7iaxnPWCOKFn6DAIOrjnhBlC4WXhHmeHwKQ9WAQkLVbC/0znIRTqJALK28glsnB8
jCmRPMm9RMBdpcW8chLi42LJ8+iVhgJBu0kvFZ9Y5wuJWnHX20jI9qrSNIEm1yX6P+w9PfdNDPh+PCfLqYp7Tbp+O
BTgdnfr2AlsDBjy3NxKxQkKP4v3pAsuHV9oabnA3HNCJ1b1Y2UXSpgrmgPR19ZcUGIk5Osu8NoX1sTUebtAJP7TH
LtEpmq7yTDtxCt7Pxfq5+3JZWNwsEuiPhdGuJ2R9bpuy9wY8bGk/YYBx4OYU3wvvyLZn4XWvjbC45/ulOrahrLKSE
rJmK4HzfUKyvC6sOhWh5/3dTjuvv3pfqSrxwkkMyCQp3Ly9vadG9edUtnZV/l/mvLAUV2s3ftb0d+Pzay5tHvxiN
nJq6Z8WRT6yIeg2PxXY9HPXPtCv9P6zGHc8d7aVkF4EDYTFmWlzZWJv4fFtgvqsmLn7ebXGb/TqpIkjTuxCQ3AVX
nkRtGp/uQKyVPHFTG6OIVazPG55gsvmnaN9U3RxQ7RqV/3u8H3vhjpjRzM6UqN2j1d4Hl2dJOS6h1kmFvi9r3Qf7K
pDzarUXDxbFGAiaQQ/EY8TzXjIcFtT8+zjqVAT08WKRgnHuuhNp/zsgzYjUSajNBOm/ZCdpf3PH+bnsx0BA5/1eDu
QbA7TVbt/nyYuqDXb5vxHUk7+CBwVuymWTOo6iKzz7wx5yQi6KGbn0QOm6kmFOyP5T6SeIs95scQZRgvAkm6wMuNVH
J3ZmAxzqmWKHXiyxoNHtQsSkBobJN+uZNpoW7WjbQ9AfGf/dDpMkUQewXuEzWS/wYOefNqLH9U5o9Hi4J6OXJ83lD
s+UvUTrT2XyCU9/a/NG8MtH1IS4n6uG/SWmvIT25vlZ6Sfl4+vWMdHY0NW+YMSD25dQrj2bm3Ysoo/NiAW1YZs/kl
LcNMx09aeP4k+XkmOEEVKySnhO6nv8rtsa84yHGdd0Mya/aDeeuyXXiaWj6pzFrU9l+fvA3ZMxxTl7w4lp+nb4v
FKMYOTmVf1sCt8rHkSw2tlVsF/xIcI9KgdBl5/a6jMb1Z2HLx3+W6OiWxiZAIcMPrdOKn4KSczKsYQIZvW1eo1RTxE
7ijtl+tV4ajxS6PO0AJzyvrwvOZCtwkL8yltrtnj2sPhqkPiVxMu4y+NHsLtdOwbc8pNQoB4ObaDbpNS0nbFfRtwRW
nKomrOYoW1qkk62aNt9aBZc0zcK0tEkmIIP4GvQxuq7OkPzE5JQpnnb6YHhE69ylDEoNGtB6kS7yRlVUwSKLYEuaT
VJzoBHoEJe8S59l5KMvDoKaTX7VuJ9mbzVbSH3HSPXp8NEN3UTVb+B3+uuPjBoUfGAOtlpN94c5SF5rBvZxgjGfwW
lb8b3LIAZFeUqQKZ24bRiC6uz+VlNSz3GcBDqWdNoxSOqwtqgAgNoWcDibgi+pQzrkDYsrlAuMdbnZKOxRawcn0V
SGcuoy8ACTRIaJPVm9OXQCgNTxrL5tVeN5dBe5EaSPmkpiLgdIaUiZkKzUsXn3Mx46kEpfeEjEeOr6vt1UWD7dbr5l
eYN7ZTqHdCbLJs7WWI3jwtHMbjF4TdwICoumZvTyntxPYHmVNNj2kWFrvx4K4L9JPk8zfqzNcWL6N2QRKGEqqrhmUl
mCAQuHXaJZ5yoMvt7NdOjEP0oESqwFXC5gjfyg1bpRSGXuYQ8dDR5pK3B3ZS/vUTx4KGMU4WMl5g1u3CHqSQnQC
OMy45eubR8Ah8KFmSTYhqx6Je7IqjJCmJVX5wIAFcvM5oKcRfJAGyZpqkVs7d7ZOeGIX9n4WDZFuJjJwyPhu5PRM
lAMrlLFxC6WjxhemsJrwzvKSh2WJiP6xzt8AyL8z94jIUnrJMcXzBo9jrnFyTrqdSWx1uQ4zmBc8oFx1HOgE0gC/
ceM9vs9r1flDFtgSypWrQ2gkpumHfR0UxULr3mWdrQT/UG8W8l0YT9AL+hv8NZqPzKlZA9AlZT8PBPl1CbThG4bp
N4LhjJb3bzLstJyRdx7t14nqzOn7vibq7IvPC2CHEfYl0bB0u1b4fyUCDdhKJAj4RdxyFnA0tchz1ZO6cE3qtVWB
OeMgkbUmKOIf9ho2KTw6Cun/xNIhv3zp2KTEk5Xr7VfWv9iv5MvEJiRd2UJL+2vlgjdEO2HmnM9ou2lEnyFaztI
wHVhjxGxA3pgf8T7Yp2LqVdRwsrg9Z2Eo7PjxrtxXeMbu1hKyuuhTYaeeRuBiaMNbpVmTRV6bXcdKuCiAkg8PWbN
3cEd+Qcp+xhnLt0skF5KxwcrUGDX7vfYs2Ha7OM6i6+5dLlk/1eJmKOeYR/Jqs53O3ZSyiWMe2+2M2JcBnRtmLeHn
cYIQnQ4NqhH7pe6xK4fqFiIG5/BfNkW75FrQKWqZ1KYdrCmrsWnsoDsYvOaI0Ezsrf/pExNwG3VJGOCS8chp0xyyX
GYH6GjJTBAH6lv5Ch8YGvW9GnD/+VsYFhXDj9WfUkqC2FtuhGLD4xCPWjnjvNuPc/FHy7c4u4vbk2vxXJnv/QaS2u
EG1dDRYSkcoWpJ6Re6x6/CS49o3VhsqBHtBz2FSJ+s8ZJWbxQDNw0HAG+3t8GX2v/RjGb7OVKCxZpZRFx9dxS2MCh
VOoBqTTNB1UbnRGCMgy/TYwP8PSMKYOLcRNFxffOBPTOh1SVV2pJemCJW0MWt1PTV7MKHOKyF1JyEnGjBq+0ZCcx+C
mRqcbxxhyAw9l/5o85NZg3tRVu7eNKTjVaMEtSuHJ4QndO7FUeYtLmq9zGjjEwJ/9ELxJ4e67DbnVo20dprO4n8bT
55aVq2lwYGMOlMsqh7nHoKkVcsGepqvPnAy66gGgbpwfaBy5bu21AN2/IQ0sZ/Wx59/9HnqjwKKzf6fyBqhFWEVqP
6U8ooHL/Vd8JLLiVgDYyGt7HK9Ppj79MuwoUYRW5W5hOnRvnrKWf0jIPu05Qw5ZiO+UY4qS+dZBJAZ+M/xvoGOxz/
cQvbH+g0uoLY2E4+zxgsnhOlcGp6lxJ1/6G8yOrlUsWT6zP6BxxOBhUEPnsA/NT9SU3iD6x3CadZh/4hSYDXhoT/D
Or+U0KU8BoEzOHHBGyd3m1G2jeHSjsUy7Mn43Xt3jRm0L5KHk/JTlfvNSt9rjHTNMTT/e6z+0U/DTJYvmHpWQtLC
mYVk9+7vw2EJvETOBE4iZ3QV3fDOFQYwXIQqoGJVuPTX5hCVHN0C9TImg97EKtsp8XhSmWmfYPQ20lRw2sdG2VBBr
FlsGLzx9yWw9SzQnVvm4yimBoxkFNwCjzGg1oezGEpBovAAyTtPKfwg57X7ZKHupfdXbGr0vR6wYJ9zOetQPUvC6
S/MBldg1R9t3WfxpEb/DEnAbJdmVO5xeXb5frcmD2aGrgtDF4/GiSiRqWiETrGNIIgSjGdoOUXi/EXPcBNkfcxwqH
NNVBdr0DPzjodcRKwWvOSlBYqs02u3LvP9sb3w/r87Rtsbv/ae0N9J6cPq+bpXebpLKQqrKhEnXvX1DGjKtfhiEHrr
H2Is9xKVi2fH7tkrZQrwb/hz/GrTb6H/Oxpu/WkX4mVrqQVJwAW6XW3k6iW2cJIBKnGs2HbdxR8XwNwGdwm2bJA
JcYv9Vr1zB89PJDXZqCEW84XL0DhoDQMtf8mH0bIiyY5xwn9BozedmXLRCw4nHH6uctGJdzzSXQuTmCSYy6Y3u1N
KqgWuo3ylKgrjXCt9m7KY9hwuAKywltdhWQKCO6ww+V9VQjH23y4GQbdmj4M7gKf+V0aKTsy4XJXWxMKVuh33TtYm
OZHQfQxgzNo8b8ft7+6ut5q7vBi8mCFqb91kLb3//lzRi+DCt4e2vGfnXyuxsZ4nZdmJ/TePIySdGSpwcmuS3y5e
3ExPrxbmzp3i58OTo/qNnjfd4ugX/gwmz96i5TIe3Pkgae9Ou9H+n+M8h+8EKEQwAAxyaWhdcRRWGz1H2eYW9EwOJx
gk6v4SnuH7OjrjuNSVO+0zvSYvF3c3k5zDgHKWZR5/Te0Y3/r3PUyB6T7a8k/PEC0X5zWedoyRN2HB8q3PvTq9tsqE4
CzdGm47tpRhvbLKKcbPqzYIYKp0dUrxXPBvLUsHYWnaNuL5PP6WFtuIYEBNS7U1cZfb7ZDcy1adVe+XW+Hyw7sxsW
B+pGhi35pfrZRoc0Rbqn/eYMR+6bV/dFudYfztrAhHkN6ZMIqkejdXLpRF4mvXr7yfWCGyd4rTTkWkR/gSgjfe8PD
HRmn3cKdiEPJ9fEX3Dpx2DfTvfZHsNoH1aFcQPHeCCK+RmpXSj3lHm/OCFJtpDAEY7VSU2unUe1SUbCoxFvD8zEn+
+KVuSDdWiBf42FWa2+8WrEY5dXJ0y+bP4AXUtzO+IPa9I3ckUOD2O+yqWOs8x1nS6oyIsY9EWUEUdIq2MDRCKG7p
XrpAQeDfHl9lHnb2pvOE+XgRoOmUsrpvTJnU9BZrmHvGHAUmgxHj9rJkdwz7pGpJj6ECtn0kFgwM9D9Hc+Nu8g2ev7
UnGQ/j6IyTHl3SELENzaXzcOcU4vHVXBMgfPaVvrLfqGkNxwqLPRRCP+84XHMC/mrXYZD4WJxAjfpfPTMrxfLG7WY
mAzD9AbSg4VvSSgIjfhkcd3D1k9T2ZCD6nx4G0zLbE4tktwz/0Mb6D7TNlxyZ7NN9LUTyHeapnU9IgD5g/x1xtHFj
nxtv8gOHYbZnyH9dGpGaITHExyeX8PGiqH5LpGx3Ejf5CLfxaOKzHU6zsyoO3yaXLJmTXuSw9NMr58xqpWlbXAKn8N
dPzurEOubVzYRtewaoGM9j6zD2kc8k2KpHJ/4aP0VSrQqA3KFH7ZDa8ZG+PYqGOMzuvcnhdeDpLR3mdZKGlG0c6lQ
f9YvW/zDqUgzkx3UUbp7MLpLj6x7wrCtCTEIZxY6MKR5zbgp6e/YwfpH4BbrJwh5i6TP5ZB6Gw31RvkYhYePK/8gn
DlUhSgugKM+uWqvPEjKbQ8atmgLm6lc25xmrvfdeIg8S3MNHpkOkdYyugLPxy5CXGRBC8aXqr6fNXQ6JOPfqFyCz
YWiDZ2+0qsn8u/m55rTXCwtCJh5UkAMDLGBfRV0+wcsGsiGb2wjydb42w/+KwxUmpbO7yn+HcP7gBQv+t/gQv0hr5
hI3XVY5o37tg42IF8IjwOw4mxnLMldHk6q+Vbxz2PvwzYj+uA25s0L3/ujRUwoTrF6QxIiaaZKC5ZAuGh3BsBSuGeY
RrSmdxjB6Fkh/yw5LJBjf4Av7k/ve+ZvfxXtX24oRztO3kQsFyDPKKAw7vtYJHhiQAAfGYXHDBl/dyLhy8AfDH1n5
LGe0uZOM1MO9fEhvDJqh6QTGxeBqxC0ybjfc3+GJL9Tl9i8C3u8uS2A6Ni6ylyjbEhHD7tdlTQ7Y9NyWOWLgG0EmP
6i2YTJhqlSFBTjlZhMEWl7XBoAWsTAGbeBtXbJOYh3qIywv2+wQeZU3SIAJCmtsJs0k7uZyrPdCKe7btbZYYFo4aw
EP+x4mxFjZ0CrKDYzB2t4jvvNJgWzQeZ2a+GPkiVDm0mIxv5ZGxjCrsiHS/FR3PpGL+uAQbzhDxW0GAUGQGZnXyAS6
vb6vm2vp+DvMa9afXhC9vA3fZPJWI7hqGKeV7uoHfr3/ScVG2z1eHiiicaCEIfuwc+aNfmOtPrhz9aj16jHoJhPKMY
Qi6P6CQ4pypinIxgIgd/+UjvtlXkq7XJmHu0aZOGw6aGfeOV/CpPN/FW/qAj8ypXUFxfyyNnU6rAYVz7NYvAkyUJwk
9M3tNDjEWZAvlcnmMfQq5Pb+mvf/nL8ZeS1cidr1O1Rsrvkfn2m3NoelRDhuVt6q7M3Fmvb8w0pgSMTkzOf3cfW2c
SwXuwo6HLXe7KyJ2VUmQTZDOjQm7d1ZpqjsScYL3jLeS1rfdx6R0ZfyX5wQK1I3C0NKtU2wPlt4OZQ6XgeJHQgM7D
KFqRmNEN5S2FCkmAOdVg33yjqPKNJ5yvd7KkgIe/FjR2/nMmzXHT6Xje4SH9o06PPZG07u+EuFNc8LdCb0A3umPy2k
pmQMLB2ppWcM6t4H+7rtN2uYKCv4PbbAx8u6ExGzJZBNDKzs1G3m2RMFR1wlzFjUzH6Qy4imMjZXyZTo4nSLBeY65G
OP+uXMF4BDngIHvkR7xuR8tc3/i48lOXn7FVgYm0UCP3YeJaAbvhMbwcAO2fUH9MULb8d2KmQZcm6l09n1Fdp2tny
Awcr0b/QqVR0y5aTvzS+1Ay5X57G2SUHuSUAf5dJ9WH9BPW1xJ2OLpqp4vmN2Y8RPz+XljVdRrkP0UrUGFq2aeZaiY
Ae0HCOmy20E8zzrjxo7Cfd5+C7MmmYgwNzMwWIfaVD85C6T/7iR8GpCv3JaXJO/d+6kP2s1x7ilnmjmHGqjjA4NQ/M
tQ38MfaQ9CrzBsQl8Ti73gH9S0Ya5qk3iwyQm3ngqvpVfAH6ALwHTkbnRFUi4tV5RebVKvrCJDuIRn6Xhou/4m8oiH
UmMMAu3vX6nexc/yCrEEyn35tZ0uFLTQzzpM9sYaEyh/fd0+SezvYzwkYa/3t0zjeQfFN1T/RjUf0+Ru5EtcU+y0TP
bKI5SUh2YiJwdhdl6PK20gpXNO0++n3F1MEbuu9XsaeqKEyQGsmLRExlB1cukSOkwenW+Dqkfl0J7Mp1u5bDD/BAR
2vmGGfXI0Ru/+qKnQKZYfYoQ+KsoFmVZ+a4gnBQDg3kpmOwbnN3wLWQ6gnEtVIe6p5YOZt0AbnP9yRmE+jC7dXEv0
UxjG0AG6tsCyl2AHaCw4ruJQnmFaTgbqM0TbXBjyz9tEr4dZlbp5ULtttYcV2QNG2URwbLRJdrUjbNf+bx3cNDAgbT
CbOHPzEbxQQMlQLUvTk6J7qb27lW4t5jGdqh5+NS2xxzk0GA8mCaCdCUXn0ebTEg3jBF4TmJcjp9FOcaUhsQvWOv
+ZWAo8Vqr2vQ96Ze95B4aOF0ri3F5sg94IbGRXJzh6iE4SSAmDD74GPuPZIGy4bjHZD2nTlsjbC6DwTJh1XqqaYTmH
zP1xNKiVD0K1LCT0l9tjeb0p1fEP0nNEB6lfCzGhmRsb8mHDrlwPZsrezqcgu2xx4lQ1a3vJkCGdZYeqzCJ/CGjmiu
1TbvcdmODJOk3cGiu4RDkLl7IgMkTXSi64qi5X2gGUcfv2GlPw0G/vqVTXB0lzRoC3cOCV6tR4AO8aTGdENuGSkV9SV
73ubrEpS/3dxCZyGaSLiHDgD0yfiDSr2jbKl4vA9UbDg3TA0pMXD2+5yK4HbBGWcwKVn2yjHDDNu8udw0NhZKb+H/
NZ4l6bZf4RP3evC5NkeNHeUzOlcYrD1K+YPBRBwudKXUGbS6w12BqCzvE8Ud1KJJ1rA/HKD2YPQGKRNSSsYuylsR
MSwz1i3SLcVqtqGHwBDTzTg4TfPm2tdnpdwCJqjAs6vi5AYp9HEg5oDJWhCVxrIhmvHU8lJmMDM+QrH9Xd4u8MKPEQe
I7FfOSxTg8IuXEgEUWg4KPBXd9Q+BcwAGESvtWbxWlmTK1wBM+R0THzXnLV/KhYHC2X/7ERLb4LU8e8ZkPnKhDtd+
gDjCB5fj0cv6kINfNqRi6qDnE2CGuoKjFXh64Hy6mm0p+T2ZpfaLJvPJwhpJU6yzx5TXsZ5+DYY91gRJ8v/1pgyKl
2QD8h3lIfyAmTrR8hWc2eZIhmS6G2iA4dxv3Nfy57qpmZ1KKkObzzQEaI2X9g9RcvbLKwKdnqi3GPzCndtSMc4WtG
tHYU8x99rn8yeucxl59J3ekvKbQU3KJf8iKHWAn8RsOpQLy9aNI2uMQiYm0vs9hJcxmCe9YNXmJgyxY51n7PJNYZwV
Ssuvc8bgvFz1ab/gJzUW+IuGUmIWQXDEoBhA11ATsvLCXwb8Kt27zN2z0V8xObIq7YSpUdYIjnm+p1VyR2zy3/w+0oN
4uF+3mV8vk1GxP84vFR0AWvLa0WJn4C6pW3k0ZG1WlcVdZPubx1/SsNcjEDMHHO/kZQOoTEq7AlYXKfsK3Ch5hvxn
TBW3M/O3TM6jF5gQkqX0o0AQT6oylkBmu2bBl3tEO7iQ20h6V9VkI3TGdjyN6R4DUq57YpNsAV6GlFr+W6xdwlYGbV
y9Ls+9ZuSZZGzZqg/ZtBoOjiRyMRrtPJLKE5xzCkQPUrqumaVlWFQKfo40f68J6nGDbs7wtD/FLbH2PqD5EK3gIQ
kjcQnF+Q4fSbJ8hfFM9HI3++1TjwoR+4YhhMm14O9vuMbMW8b+GJrkHR7rOg3dYbDQ/xvLMsG+Da/BTsF1lb0A8gH
qdPpVz3sn8mkbzV5jo36ZqDI1XNqXAoOcpP/HR9SsK9E4jt6SNDYj2wCsrsiZLAd/qnUrf+hLe+qQoY2ATz2qzN6V
IykHcqMfdLwZ2PdPzKTKb8YxT1K2YvX9k0ciLWwWl/2rSxjE7cZqHBvrntMpiJf1Pc2jc/o0eV6BHKTcCyOdr5Oc
3r8qLxCQa+1tnxpoxRxzLb1aDBtbtpY8l1wXdD1sXLHvvzHaXOkZPUWl1HGnYWa1JtnNRc3BrBmXfOzXUH2Mt7PxzgV
4e5cDzPBhZtLzx06lbdbNc2cL0npyjyzrHjNx3/WRuhXIuYwVmC5/oHKZjlIVp8+YXVODRMgWuWhtjAgjojqXyoG
DgzzHZbh9yI+rKK/n4i30YMt9AQ4u2FqtC3m0ZmNXbMSyAqf8Kh9Tu/3RRcz4bq89TKxvXw7hi6NJEVL7Hd+bNN+h
2fF+u/V+YT3zI+83AbF3mbtYh/I8xe3GStbupHWPhvM9MhH6mYPngmBb6YVLh5PV1Z9h1SfgRI/LYFB7/j5cOhQR4v
uE1OGtz1NZZu1EEKUlHTVngG2YZ2X2Zi8+1l7pdzsp/A2ib/e2ehpkPL9uw+ehhwgO2UVrfqvQrT3u5z2P6Gw9s4Wbw
S7TLBtjPxLNpS8vM4atvQHtRzR0Qf6p6cr/HV15Sfe3e4XAvW9M9rGNTM0rUa8n1ZVtP2FqV3i0ah6OzHPFjTYP2yV
PZbI6GsgB11+8Hy/GnHiwZsf2hlaBG1yd6NEgsiTCtSrO5NXYcb57jdxvbMUA0xlzF8yNbQlQzAgkkaxj6NOgkrMQr
CXU8tTW0X1Dk+EiF9GJ4iTVvAyh+/xHbJB3S5U2XfwjnTGP51BNMeGjzmYRjeTfgvPnOe2AufBvBXRvoWl7Q11VWb
6KNuhsTu2xjDe94Fxq6+WXF2IRE/ArErLqCGdAMqE2hI/XXcflrnZZ7D4M1zBcLlwH7eNlBAiiivNEMRvzpq3kcbfg
u5WqS3BkjrRNOROp+R09+FmdXxNLT+3BrwWbxWDgLpNbf8dFzhDJjuyZJpH2uhlSX05DQOntiefAUQk9VIkZDy5eov
a+aOZdRvgOWYrpVb5bdwb0jBG9IkLu9HNy6Kl95Fqn4C6h85uiwzmbBRbfZa8ym4b3ZUqntmC6L/AbmBpjCCxcL3KB
JRf7w/o5KXWdCfZDXa3anymaU6t6IoicHC9wrOH3W1B6W2MHn28bU4Rve1vvPKoOBuWncsKsEr8JuN7Py8qxJKoaH
jqOCYrxTVRNlnmy8cKsyQzH/IKYuEjDCZXh0B6OZSt6i+kqUPgEZNYWuHxLImsxnCjZU5gc9X4GT5G1Wzbty8L4gZm
0Cl2CcdjC6lr7j7E9IlUBQhnpC5vnd6x0eyWhbUzDfyS9sc8m2t5GZkbQSHBL/2GK6Yw5RWvqOZ9ZEYXlyiNgQ5jn
P/6VIEfnefGdZPPc/CuqSbcjdvN6JCBGvRpwUwjD75EOyMwc8fiKY+S91YU4dv3Gny8SvJbIozfyFk1qZJI0OxhvQ
bvJzDtXDRGFv1e6fM4JoFOHGl0rOgqWA9iEETdnVm+0Fz4qibYxUlVY9Km/dnXk8UznuhB8Ms8hN/HYvIxhl2MkaN
eC8/JCsB2sLKjIZpPSsCRTP4RA7ixhQzJAMgJqSu8+9DAY9gc5skErGLc8qeuu/MRGG96Mcvvuu6lLCzFTWpj8Sgp3
kFvbu/qd+85UyA+Pz6erybX5y8wfSj5PZAvOi8h/R/Wv1xpUa0z6PW2/v4Y7iQzmxi5F0qeFhUJZs11b1mLrOuKWn
x5nbEsDc+WrL9dQFHTzT3YoQjNBDHJzB0HyoT1EN/wwd0o70jI+Mf+3ZWvuDHC6+/XCxi1WQUi8nP97ma4ZPkI+7O
5MvjLqHmn2WUdHFngULqKAzLv5qrHRfFWwkx7Qx1/XL6ZzKgPbgErkJyMRsygfzeLoAlRL+tDFCf/1LBIuPFzInGJ
XTIdaw8tTPd8wOOb9kpJvh7GHZEsZo2bAGlqKyBNCSzcYGeQEXp60w/ovOSn+iBRk8+ar8iZmZfeW6e3PR2x4xK9
ZOtvuYjZu2ELs0KrvpvaKhULbhXvlYWaCcOPCgdzcuKBI+6y2G4UA0qeFxS8jWnK6OwT7y99Hu+ZpSB79Y2gpDxwDT
FRtwaiM5BTM/C8qDH4dD1E5T3TvG3J5gsOrB1G2glo7SKO9WFYtMOkcWgTYl77fUFYGXNZYy8pDCOGyp3gFe5KRaI
GQgKTI1Twz4gXSHdGQPGChHRu/oZnXQKdJFHb1Y3bqHJ0ZqvAm+HwYfEnIJTvzMwIP8bDOGWNhmZ7kQQGfaRhAfV9
Fo4PJIXzH48swGGc7x7I5UnD3Fcb1AFrpd/dYNWBgJQSURd6kFhQe/5qMIk6Vha4UydEezf4+vRxkvNJq4c8x1PqMS
i2tS6v17Et7J7Og7+7+gdlx7jsG4tymAkeQhyr3Ba+4ERSF/FzjuDQeoIqzJ8yj+bx7bQCnhtWxPGXP+dE58tZcel
KDXn/fIutXGXbJupD6CJlp9z3VJWjBG4SFM5t8qlHNpiNcKHCm5k1Nn1HhqXGNSqJhyxq4J5TVPs8ttjczuXyQNQl
C1eSOZn9YzTBzo1vN1L4GBRWGK2g3RnD4aAXhl1Vsd8ZR3Xn6TWnpRCzPEFt4BLuMXLNsFKDhVqDEI/rEHQfG6BEWE
ltVdGFNadiOjjHPGo4UzrgcQ0GM/B2QyTGB4A6EhM8ZVqcfB5BwqYqUgQ5h5Vw6L/xYkRydr59DdtIEIwgGLdS5RMq
lmp/wxU1yOvDnGwSLZD8FaL3zsqkqE9cTEV5rLm57dJh7r0eXv9vz56zZoMncjxWPEGl5SyN2QXqbSXy8Cjps2yGHm
0Tgq3jZwN8O4LchUo7PgROYf8AdwUzdCqHBl9uA/2AEr8LnJBvqYp1AxJzVPa6qX7Rcie3TT0CcjaASu2ZsLrZkHW
dOVy8fZRg98qRCQCChFXoe4Xaxrb7kJLxS4cf5dTuJz4ZZ5d6Z4vd2nHmQM3AM3tw8YFCA0JjmbGg0q13NDmdB62ll
CMQRP2PrT3KbmqBJ6S+3RaPUfdRpNnJ7UJdOsLHmPYbn9e4YhPG0ffPiPg8KkZWEgA5jXvQ4WBkNxlDZwPRxmKs4P
QgnDI0+2aoOmUhYQRnCYtIqOFdQgDd1+k6wNF4MiMbXuRi5qBnA6UeoOMkNw6zbtdYlwFVcfEnr4FJcvnXzJ5mzffZ
8kCUIhKxvy0nGNkfQPsmqu/gz9cpExRfEk1XJI3E4nvJ5eXoRmnsfjfcVzYXv9Bu6RFuq4L2u8/tBhHzT5MNdBqUt
kcWqDG/FRYbJ8N9/xGKL8v0FVJ7CHGpddaNtZnXrFJ5+kBcWWRdFZ1Am2x6h3ywkHmfjxrpiDknr9RmvK9nvKtBM
N7LEMUbf8zU+bEzxsdRvt3Qpgn2T2j8jUYj96U9k5r8hEH+ANd3xN6OMdNPVM1ELemx5Trb0cxSvPmAy83PnGIKUR
Vgi6PHKzJjY2W2Eq9x/hgGmZRkPr7nqMaBuzcfgDyZ0PQqE8H7iOgfAtZgZ3qzfHDTN/hMVRu7ynlrf12aIEUeTbw
Lhdd4IgWhVMVGo1dDKBbEMWAvSOqZLjGgGxXCZkNHWMGVoG3F/n6d8G+8V3FtOXqlH7d09TvLqTrIvyRXm3g6nLQ3Rd
3vUCJ7Qe/4QzxqeFFX8TaqoRAebfzzDKxQmCUNjhG1y/jz42GYrQGdGlPHUm4gRv6Tv3ovTGmWRUnbXTdBD0uhGDIk
JVrxK/2Crs+cn87k4hFTMc9yo6ZgsoP5Ubo0aDCa0ujwJpa9wqRq9RwJkeXVk8DQeHvF2uOgCjB5IrRpW9zS+fZN0nY
wQiZUN/ivuJxmXyFLB+D5ae14FyWz5cVoBGoMWwjUoaWgiMs5V8OB/b60rGD9KoxgdEqHiRmGsgLGcIOQBozPaq6bl
ZiwaRJ3oZ7W/P7TgaAY6sL9x2l81gLfQYWk3PU1qVTlRHCBAb7PD5jM4eVlX0Q/xvAJ7oq8sRGbyKzoaWIOsDRXI
kUtr/oLbeNL/jN2QfB9Iy06v8Eu2I7hqFtOWwGjG3a5vXYtBVOYDLT9n0ha7BkA4g2dqlhhp6X6PO6T5GTfZXE1Ws
Lf42GlTGXmmKcM9Sb5XDeUefmaCEOnPJ6mJcPm3mzcgkUmbT8QNbf0Q3glMdLwUHQU48RKFhEwbD4ZtnOPXT/Q2cK
kaGPFNO3cTvp6TqPpMCcyWgTpUz2MR7L4LXcS1EUvVYN0CmCmmBjVqtJhBkR0PiQXtnM7XlsTP6Ny3kXpIC1ovVmC
jPqkTeN2H8O0IhrnsXmxH4mu9/4xX6MzBJFE3b+pV5ZeLfGEqXAnHGpMhWt0nVJ5Zd4m6OCtcKSgW/YTCOfNE3Lvj
vqmMxvRE3Nph3T46v2E6P2E6EpTfy9mDjbdnQn8fRDC5YsvTbGXL1POlpeaWs5ChlQxKY2THmODkrJ3sdDm2ngaP
3Mhq2KTrSVSYJL+d4Jq11HDP58nzkscIKW0Gs58ooUaNQLg6HQbE3s9v7Or4Ra6LAvo1JNTWvnBmAJC0yDo+yWu35T
OZvtjk23nwSqK3Xw9SNRbLZnfDIVJSavOLmlyUjxSW4v4WsmjuGkYf/xiIVCeFwq6FIdk8XIKnS2fZA+zDJOexfmM
FfWJfnh6F6Y0bdpvcnQo1i1b7KbSEvNxe/r5JSbaXMXkfB4Ew4rQoKPfdEzxgl3W63kuwr2FnjW4dcxMDwz8yUmk0
eIa84zdD9Ofxk0t7OnXeHfKBBzB3Lf8p8Aos5q3sBkX1SrsbAqJM0bxqaqBAqcbN8dYVKpxuhXqdAUoxfRLqbmiGdj
nwaHBW1gMn2yA5FuqdRAJ2YNxAUuu4KAYWq/74VTrXcPFQvtKOcn52obExTzHPXCMg+0gfo6C9tN34QfIzQsTgO3
yQUYb4hYRQ8kC1Ks2gHgdNFRnTj4tbl1GbfhvGiGcExX4gW0HCJFvQg4FZlwbI6pj5RpgPeDlojg7swojw87oMPHqX
gDRcFdS0wTzEdABH6B6eu4PgwTgCeaeg9G7bbrXHTJowNq/5QNlwT1aDa2dzsNPKO7rpsfm9eatrYsOvYBAJxWyX9
ENjxWvOwOHKIfRLZNWSc4Sx36D7Hti0zbj9Hrj+ub3XuzxvBsH9nKwKnVDQKchbBDDJ1ErBRwrOQ6rrBpVx9JOp9Z
4MjDjTOuHhNTAR8uASA3MS7/uqrXehyQPzVWbdcVnFKoztfVhb9OkoD/AMAxpTOTkxH0Ov4tL4xsRk80DrTGq3pwK
lnl4zMRQGyHNEJYm2cOGBSCdqMcN6/D6HUDn3VPEWBp9ZYFQYXOjlFk5y6NV7BZJrLmaTRuo5f4HkQs+Xg4zK8Rh1f
Gc8Em9UKeEl4i6Ul5IlJP9CtZHbWmJxMCrzH6xdsBq8cIzTYS6n7Y0WrSxaxIpYdn0V88LKlJGHBakbu9SJ9nF3y
bqWhqaMqM9aU1X4KHsXgS4dYPn5ZvTmMkDJUEVJVqNqDfbfhLxTUHBr6FHrKy4hX9OSuXC+XMIyvlNaXGVyKaCqBvt
8ZxQS14Se71azTLGUcVWfNo28BpIPsn3wXVMu9ZvbMKSh6o7hVdkbzJ8qoMG6Eea5lLq8oI+V7A9CP2aPzcA/LOz8D
JnWrObO4Cs/B3HHxZKt3EIUXt4X9Y2TpGuZQeBbEjXpX8RTaclbXhUYO+V3bvZdkZjzl5eMKDlUZHZUb3S2ctVdsC
V3r35e07rgMFJ8BwjnLLFOHC/p8iQVbxQ9nTGJLs2wuJhRmt3+0/2HYHAR4+XcANx6o4W9GDfcoDTcFrNPyBRSlS5S
v9ocY0ZTuzE5NUGRSBDJlRcZMmc0aJDnlqpCkYHV+6RdG1oJPQbs8w0dZjCRFH5QWTH2AtupEldq5zMrpa6gNxFhc
PmFjm5yUScdJ1SQm6chc0fsxs7/2w/DJUWUiTtCLJUDAdEi6rSBYzkYJQ3N0Ob3azSTSIF6h4Al76BjxOQeoeqS
0yoAeo9BcrONSC0ac16wWqWqqPhGYL5GknfHoxZo2VXNxKvVSzvA4X7cmm3g5WnCoLCBOOoKU9gzRJrR6DTCF9Vr3E
FvVJMxAwOMGrK8GRXoaJ+rNMPT5n+OBNEFPEzRoQ8ScSF8uS0z7JGRvCVIoNmTJaGxDeNGSSXxOYQH0HL6gWzKh0oK
mN7q6Dm6fYyy88iMF6b+vD3dB9wDmTdvb4eN7y6ITG4o0QqkrvFOsnbOm3D0SgJ49gtr1hWYdEP3WQ2P2yGGhFTRA
hDg8Cqe/OBcp6VyXnSEG5sJjsuVXdiyYyrcbEZzLYUqlsWI2+KHHz1vYK+OJqJ3PDcbORvcsNUv0P4+UFDtW2cZAg
FhW14fuPy9tbqYSU8oipQJDYr2TcCpOpcKtxmm/NXiLLVdEySHwE47LFvDkXDUMsfVnRjbJ5TLqNsM9Nt8rT/xhPTt
8cnN6eA3dZnQoJcUGKVXcQQABATmUfF4AMALSLS1hVbQ4Fxzm8FJmmLOgsk2gfoQ9gpuEWrHnCqY6I/x5AaJMyBEu
bmZIQEkcxzrpOnMQr5DpUE89gSRlKk/tbq4z6ycWSjOWhOOGXWW5RN4/T55o9zs6pXlQw1TwXsCU/W9qaBeMMyqmlq
rH/5qLuqzj/t0p7ViAPmP+3vhkXJjUQgVyUtKcnhPv2kgHn88KnHdWlY8K8ZUJ8XtsnRT9Ay8o0R1/j3AqWsN9HGK
jZZUpg1Pdix/W9J4kkGpyOzrXcjJ3emYxJmE6f+R40KZdDHAhsTZRQBM8Wa+dnuUPvOoIlowMoOqJmJqtl5TvqIZOM
tybhbdpQF1HJXS/iJDfzmtbIqDbqIK8CPQwjFmZPxFqftzSClk5+sRI7mYWOMnvZfyVNTtMdr9QdVWP2dNMfh8RzXG
ZUEDGffK7iOCsIufFs0PqGPK2+TwhpVr+TFqXG5mI3Wy1VxH5EY6Lc7LClyqVlQHKNaMB2wH3I13DMXGqP3K1xzqcG
HQ5trr7rmzlhFkDEOsaGq3z+fM3Ak1PTCZWyYeV4U+LGJwjGMLkpxcnN63Jjp3GcNujlYJbdrxp7DUyTb9Ps0ubJBg
5hrg14lGRqsm0sqmzYe4V+dC/hs1nzExg0e9f9k0ZLVRKUdhqfG1gP/BgbVyYq27LdUKojfNHKPWnVnjY7cjyXPJ5
UfOTFkHY7IXc+jt6E9NyRunOjPVmFUXSfGVozxQMZ6gbr8eKHvsfjK9t6ccfaXf04u9hVh5l2m97NxhoG4bmL3RN8
OeoYnaDKZHEtUGCgzTFLFq00t7C0/ru3Yd8PIiajV9lDhPJtuTUbOk0MsnjoIdA2XGTu00zKt6RFbGBcQN7t1qC9n
PbcqS+IjLQQHOkQpz17GVP/hLSg8cIBOdgLGlG6+lp3N5tcNzXR6C1Zkl5O7RD/2BdNHH+N/XmzkoVPsAhs6hUXSxU
XHyiiYy7tvw1RJsRzWNOwsBzMUxZ2TrlzdQxBdjd7sSkYzMLf7G99SIBq+mYHN7n6BrSrfV7L8SqQPDWHjRGJH1
fSFDrQW4cQ8lNUJX3VIItymMeC+2mV6vHE9oUm/W+E3+fkSi1L4JXTt9wgwH5L3RmlnuKY/fFzXmvvKDW4jCIkTxdq
HdOSNUKgsgdQlNGcHVy9rc8eUglg/1NQKYr7kSgjFmYGCFoE7y6yg6N3uYnIgeMD2f7JKhEgc3/qMxhUmFeJxdHDU5
fzkEywTTDlGSeHbUOTfzt8N1bQ0hGXmm4btgC1bqqyATZsTrOOtE5Gl1svYUn1dpTIxI6z3gqw1gzrmvZ+3vH6OB3J
z6Pu09CFjHe8h/kzjxENFRmklMH4kHI2j7k0xazF27RzO5l4lKM6nojGhwyB4sFgOPl0/4b6SR09w4thM3VA0g/V
hrn54K1uZjXFsBz3w3GB5+BO4oPnVKBFNclCcF6bOHrI6yAdHs0EtN6H7hP0KDJUbOQFn08S2qDTXQhs2Kn6g6eXu
9j6GK53Nu6g4gXGT7UEzHMbZCh7AdKrDpiqdj6w3t0Y57kjm+aHm4xo2v+Kn3F+t4LqNJP4qAWE8JEsUtBljdmM2Fz
cgiX6xoVT2tjRXA3Qq+2Xa2xcz+LqHvvHlZ8Bc8AFlrlJhpqJDg6HLsGCd/0xSSu0Z3VtXU3MJNlNAmx7YsOTPqpq
8aN2+xhjQlB9gC6UN/6nFmIfC+5QGu2CAexjXswzvFlbDOAaCbbvI/qHhQoVLzDMdLbUZNoEAn3yjZf2s/TrtIq81q
iuqV7gfgJgV8Q+T4N08K4cLdF2F9jWXCmBNnHz4OU9X/A5VM8bAGu0S5aqzXqsfZ9aQ6rmsu6LOsYV0ja5kvV0aUhw
NFL9qGNHMlLzM0tKJZwIbCv7Ih6XnO9eHEOnqG8Yl5zMGKKxWwHxrRJKl7hV9pscNQyI3FRdEIvfjO23vX8t8tJ8w
6iPqAknDSwTVrOqHHbRN4vFRIlxBb+HzeSbNyiBD6PtRqGu4gZYtqb5pSIrh8WJeUqtLk2CGYI0fQpF2T5sNp4qCL3
0wY2OsljFTFCwNRMHFUmc/VWHu6Tn3GpDnSzC1c64cjPqD8bvbHsBJqgyHp7lyu9DMDMK3eSGBcqXoQfQ97IaVCP8
PHg4nlWCGvY9M3m2XRVYDuKGVrP9oK3B2XoiBzVRCaaFYz8mBUMAHbpn0hT7i6hq1y5KWmNszIaPgRKJhNxSQm9ox
FZBRDk0cXGLLUIO2SRMOhr0c3QPkvOhpvdngyPnjm2VfkJlYZWXQLIUKc2ZhBTIK5RTEx8Y9ESyOkXeNjf7MKfDx4
dcRP4LNdUkCMuYY5Vo2fsIt8LvaOFbW95J5hS2B1kvtNaXeeXz5pPAU7f2WQ03J0+4KdLHQJJ5h92G7b5TaTlBFmb
sqI2dYQwmJrHUI4Y5oOzfUP/9vNMPr96P5+/Qm+kIrNCvXYrXWEGgvOVfrzKpBt40Nn/V0T4XKzBpuLL7JNWDl6ce
x6pqpfyD+2JV5xNwDLRwRZ3sr1nHmpj1qDITF5wVr1c7o8fsvKBcEJLtcSGSXx6cTjnb5QAmDgymzBgwL4g2XVRjV
SezWMJzwJjlLnJAn+ufMtnZaHVuOC1CSl9JYKgZbZy3RfFhqM4pK8h1KHsSaXY93m3PaW6J+nOmPHb4Ie3x4Uq/PHX
AiGK0L2RTAsBNkD2LGDS0HDZS8o+5pIHhvJoIpPkuijwn+dOwpWzVqBl0xl05+tlVeL5rQzEyhz9JxeI76xhqqdZf0A
9+kqkAlvdEKGUdh9l2kWdXJnPO1C2SDtdV7yUdyf4YJA+IXH/ZGm2R9bMaZG7mqUmdAXRYZX7JkIX4tdV1fmIydjB
rwkJEVHCoKuUKW4nRDvM3vQ4AxNOPDsbEp9a8Hdcgt1u0KnBLcqXCNeVVZBnsvXKsjFL4eLPg3h6wNHNSmoxOWtsS
1yo4VIIMS8vHtVYwqi1Yqt6L1sEjYIxh/w5YbqFZVUPYzQvWsdfDR3+AT2S2zvi53g8UA7lAZyn89o1ZHm5/kYh4N
V88A9yOcTiPZAFbXhsAPvdHozT6xJfT8LKTSYYJ9VtbXdq1jtaDk6SLdUWRRS3ICbF9A0SIdjTmBz+rv6h8aDLdrm
n8zDB4PRLpxj0Xe6Jq4nHLxqBDlIrsbEmDvjXe59l1r2fUZITPqvt8PBVQ5HTx0JqNxrv3vLSTLfrb0e4V9Vj2xNB
4P7hSUk10nW7AAjwYn09wLvxcV1WbAaXJt1dY+SNIeTmJDjCgIJvLkD9JqX1d3ck5VBM8Bp9PuxXNk4jIq4qLFs9
nw12b0XVbmQm9a5JPqJOyUrN3ZmIxGhrV3E1gIRjZUNp6BqXKUNngU+oE4s2WMV7Lm+8iIyf+ovkHXRjPGxqeHeYyr
oYlNcQlkR7K3/Ab8mMaT6CZXBNqjVJ2NY9/tjq0Y9X7r1r7I5nc0GZuC1f2uM9H31XuU9KuHLjwjFxxK2c/9X8vUbN
c1E3DvfpQ/TEH03hyRk1eMkg1mNn3tZiXlFwUZ2WR/RSxVbHTm5o8xNBDqeC2WvRGpZUJnM8YQRGIBqE1HfZ1SXTsm
8kU1aFw8uz4kaG/n5p9KSTiPHrh1J4Cc/NXSgvPGd+AqC4zdGrlwIMnt36zC6AYAzgO/WVuh2Sc8jM0VHlqCm5Ro5m
Bg4YC8u3H4Tc8MEKsfqjuFw0hF5wScAIrB1a9MkZzCYyQhDbr8N6bXgGKLAaT7UO7pVJmGuWuWpYzkZr2fU2sMDqM
LoOCaq3ZWJbTSVHRIVPzKEIX2eBsX8oo7/xI0CQH4tPRzNdN8jrRv5kMEgDx3VCTAQrWVHtVpmojJ8oRTezRrfUvoLd
Wy3Lx+3AK4MoV5RxmjyTavK/UuoWXnmc8aXd5pzWNxmkiz89ytAfNzIvQxf9RoftQrAKSyUn+dx4hP/mJyOYHBjsEV
jt0qFWJ7Vnm8k8TyBuQcrpr1tWp0XLQajGd1JxMHGM/LmVUYNfxTrrrbKXXvQq18QMN4W3HqR08vNfyDm7ssfl0Zfrwu
8hbvBpQT3/+l6iKZ3/SFxQqLjqhoQTjtjmeKIAWYgEfbdDo5UbExzU+nsmPdqfKlqs8r/jOv2DG0bn3LWZHv7yHdM
DhKv6EVTKGDnGW6tLKqxFTFZ6RVMMOa1G1cVGVh5C2gqJsy9Wzj24CHYwGFS/yBP4yP7Xt4S0HuNeXm5KeKGzRoCG
wdEH0ZvHCQdPzXw5Rz8j8FIVPLgmc8s2C1vPD0aGGj5gITo3P8z2jh4mpSGJ5m3XG0lbBNYR0BMrVOKRMfNEjZuHxR
J8R5Ti3FOdiu01F7MgMmUu8N+L2n6C20Y8I6njDXjM1a8lrZ3E6jTlzdWJ2XhvQDprFQBBukVDqEzWxl3dJmC6a7a
YkoXR+VDsN8K+2FFOAE6+nLmHu1YgsKOm1YLdo1oRb+vlEJ3HBlKO2rgJWiUvkEZdjv2Hm1yRHBlEWqzn1gvL+6pGt
AZTH/uJt3pSEvNq/j5wyOl9iO5Wh0CWlUIqCbFxD5jMIwoLM1x5YJvwVcHJuR+ZTv4qcJqDWPCMaqELcaeE+tFRY
fpfTFDZHWTyQRVfzF6HPxpCQIBxG6lL4vSVA06B4+pWwPQPT7fd9R5Q9qSwLpwqcF7PxGPb8HVJx+gzOd3GY4l7hnAg
bJ3afYCcRBkxRKt1zYfQpXjwxXyJs3nmoWDUSBc8I3VaiT8lhqcx/Pj6LQDM2J08BPJ0jOj8Q2ZZ1tq4RsbgAn0JVq
SrHS0E2Z5u8+7OjszGwFcEkOfBFqYW8pM+aWhbY7MphlDYf1b/ioP0nWjPdRQ1ba4LHkMwEoK1OiQcKBi46ntUO0LDh
4jBZp3A3WJW6ONtjV0J97tlS8Uco/lBnW5OPVxGxkNvkYvsaxaZs+5YY8gLV3/GMuPy3dgLrvXrSNQLJLefnfxCBn
8y6WvTUNwNsa9QGqr4DIUh7ss3iLPtJxkkgCoEyRKpfSmPa9XCP2dwmzYAa4m4iDNGcgS5VV+0ha8Yo6f9FwU8e3AO
iLHDSmLMlZ9nnWPaA5SaIfH6Et0mczzyB2AGa7kknlPHxaXXB7cUX5cIFxVQdU3ib9D68vOGVVGxHHDmXgHs0I7Uml
ZOnhH3+v+FsO7FzbuWWLt/+aVvXf4o5kNjfGtn30B0OUtk5kYvJ0ERvIDrHz+ko3oHyIPVEyvMHdCjnx5nBWvWg9N
WkVxeUbOzg8yIz3FNzgk2hm4KEGVwcnEjcD4GbiGBMRfJGtnqjDjKzQJ9Tu30Wjzd7Sr6/tvxVOC7vCsV9H47ZZ+s
fXHZP2umK9/KDNEs7L3Hv3z+t4wUB+fLTofYxU0YH5v37E3O6pCwE0/Tez6cFvJZl7cIvQaGnIKHxcpcPP0g43Cs
XKYGJq/zzXpY4FGT6T8oBUVf4TydOxX2EQ1zHB3RXDdJQq0mLl7d8M/OH7XdmnFz80NUQ8C5nA0/hTpNhIk8H2ZRcO
UMnO1gV0vWPWrlvVGCcmoCJeP0d94V7Pz+XQL3eqD8e1YixTM3MbJxq0ccQ1rQjs3D9d8B9n+fO/RmPLY8K8Apupm
dlxlrhYX5PW6sVnxfVjo9dHe+rKDFfj5x0q8dxQ+4kMvVX9oGt6oe1J+NGD6SHbxl6zLYeBuLqjfa2MCa+BRxI5B0
qYbpdR5ZCslRFcVyO9eQAIPVnLpbfa6DNH2KvG0dPX0C2/Bvs3QKIEM8KdgGB2UBCz6+fFb7YkN9Kt4ktjpsaZLHX
Xi7Ma5WhpF7aHdT2BvGDjlFGSf9pa6rl1yIfGcvOb3hZTpaUErjbDpiTi62RFDkBkc8Ii/3e34yzF5uYbhnfiRtpLS
r0OYBrBAzx8ThkLwB3G/ZoAsuhlaUP6WDrvqnJ6PxWplbkgsCWoDzj8tsyqQd8n6ZUpU4JONBEUpuH9kXY2KT63d6
Oy4uHdGFuq1d4RHYMt2j6EQYZcFxzdW5VlZvrdFj9pX5Jfbx0hGYAUBHEVWT9fRbBIbZTb1jKRuHiA6Btlq2KqWw2r
RRJYAN3C1sHX0Vk4GrqNH/QGbPyjD6YSYLM9FC2iFzgzK0umw2gX6TjCY1y1dmrwpM9cfFMqKeNzc6aE9r9M9h4oE
KRu0YtGxPXrN5cfyJoUV7EPx4+mUCJShHtfzKNyB+eUDJ0a7D1b2ymC7Dn5QKcgoE/HEIiW6gJ/2LfSUHAg6ekt1C
4I5f/4Vm3NCFD5qYnNX2f0vDjBD6kn7wBwoE0BsqdIvAHqC4dhDGG2XpqAvJKlqyHgrOvJnqnxrnwK3FLX9x9UaeCm
aYH5GXsHqKCvTrZOp6Gn5fuZk8jWQ+lSK9J2OK1e08eZ7ZbUuTX2KiZ3AaJMsNS1mI2PrxHGh+E4P0VqlvQ1L2rGme
kIcmRONcNFzk1rWDXpvF+XTL5mtmmXXOrCgOLgvT9cXXA3XUXRD6wxHDrcI5Z1tj/26aGhtRHzCXMJeFYrs+Xp6Gng
3BhxDjvY6tsqV0UJkkIh5qG+Id03WJgWXBJmNe6XG1+7z9V3mDkXS3ALYr6DpLkeuqE60FwrKiEjbL0iUWmymz5mb
Y4PEFzUQ9QypXEtjzWPsaG6UMhNa4zJ1tNgRTnOwTNhnPwJZYEZymMkmzH4DG0N7Wo1IUQGqqUUHTLodqCO5Y1fg1
…"

Qr776KmeddRZut3sARpt6mTBH0DyHk0yYI0B1dTXHHnvsgP/ezNS1oj3Hqz/Cue5/eTR8HstHXs2T15zUq9dnws9
kJswRNM/hJBPmCForBptxYAOupz7DfrOIbRf/lUXTRvb4tZnwM5kJcwTNczjJhDlCcteKXpfvtef3+5k1axYffvg
hn/70pwHrrsXo0aPj51RWVsbvcpSWlhIMBqmpqUm4q1FZWcnChQu7/Byv14vX6+1wfMSIEeTl5fVnCj1TPxK8Bnh
N/Nl5OLwBvDl5FBUV9fqtQqEQ2dnZFBUVDdsf0kyYI2iew0kmzLG9wS5PyJi1IuFDi8FrUOiMYnj9vV4vMuFnMhP
mCJrncJIJc2xPa8UgCZWA1yBghvHl5vdqvciEn8lMmCNonsNJJsyxvWSsFf0q/gsEArz33nuMHj2aSZMmUVpayoo
VK+LPB4NBVq5cGV8Y5s+fj9vtTjinvLycLVu2dLt4pJzTXrgiQTxO648sFImmcEAiIkNHxqwV7bl9AGQT0HohItI
DGblWQHy9yCKk9UJEMlKvMqVuvvlmLrroIsaPH09lZSV33HEH9fX1XHnllRiGwQ033MCdd97J1KlTmTp1KnfeeSf
Z2dlcfvnlAOTn53PNNddw0003UVRUxIgRI7j55puZNWsWixcvHpAJJoXTjnCGg7hddlAqrEVDRKQzGbtWtOf2A+A
zAoSjZooHIyKSfrRW2FxZAGQRJBTReiEimadXQal9+/bxxS9+kcOHDzNy5EhOOeUU1qxZw4QJEwC49dZbaWlp4br
rrqOmpoYFCxawfPnyhDrD+++/H5fLxWWXXUZLSwvnnHMOjz/+OE6nM7kzSyZXLFMqEM+UCupOhohIpzJ2rWjPvvP
tI0BQNzFERDrQWmGz1wu3ESEcCqZ4MCIig69XQamnn3662+cNw2DZsmUsW7asy3OysrJ48MEHefDBB3vz0anVafm
e7mSIDEWRSIRQKJRwLBQK4XK5aG1tJRKJpGhk/ed2u9PiH+IZu1a057EypbIJEI4qKCUy1GitGHhaK2x2phSAGWp
J4UBEpLe0ViRHvxqdZ4yE8j2rkVdAd75FhhTTNKmoqKC2trbT50pLS9m7d++gN3ZNtoKCAkpLS4f8PIY8dzYAPiN
IWDcxRIYMrRUy6NoFpaKh1hQORER6SmtFciko1RMJ5XtWtFCNCEWGltjCMWrUKLKzsxN+uUajURobG8nJycHh6Nf
+DyljmibNzc1UVlYCJOxWJCnQvnxP64XIkKG1Qgadw0HI8OA2gxBqTvVoRKQHtFYkl4JSPeH0WF/NKB6nlYKnRuc
iQ0ckEokvHJ1ttRyNRgkGg2RlZQ3ZxQPA57MCIZWVlYwaNSotyjMyll2+5yOgTCmRIUJrhaRKLChlKlNKJO1prUi
+ofunNJhiQSnAZ1jBKN35Fhk6YrXe2dnZKR7JwIvN8cj6dhlkdqZUthFQZq3IEKG1QlIl7LCrMhSUEkl7WiuST0G
pnoiV7wE+IwyofE9kKBrqdd09kQlzHBLsnlJZBAlHTUxT2VIiQ0Um/B7NhDkOJZFYUCqsoJTIUJEJv0cHa44KSvW
EwwVYfyFehxWUCoZ1gSEiIl2I777XCpiEo1ozRESkc2F7p28joqCUiGQeBaV6wjDiJXxeh9VTSuV7IiLSJbt8z2m
YeAkpu1ZERLoUz5QKtaR2ICIiKaCgVE/ZJXxZsfI9NToXkUHw+uuvc9FFF1FWVoZhGPzpT39K9ZCkJ9xtfQayCBJ
Ss3MRGUBaK4a2iDMLAIcypURkAKXrWqGgVE/ZmVJZhtXkS5lSIjIYmpqamDNnDg899FCqhyK94XRjOtwAZKNm5yI
ysLRWDG2xTClDjc5FZACl61rhSvUAhgw7KOUxrPI9XWCIyGA477zzOO+881I9DOkDw5MNrXX4jABhZUqJyADSWjG
0RV12plRUQSkRGTjpulYoKNVTLjsohZ0ppfI9kSHNNE1aQlaQORqN0hKM4AqGcTgGPoHU53ZmxI4dGc9tB6WUKSU
yZGmtkMEQjZfvBVI8EhHpC60V/aOgVE/Zu2J4CQFule+JDHEtoQjT/+NvKfnsbT88l2yPfv0Oe3ZfKZXviQxdWit
kMETt6wynekqJDElaK/pHPaV6yt7e22tau2LoAkNERLrlsYJSPiNAOKryPRER6ZzpsndsVaaUiGSgoR1SG0zeHOt
LpAXIIxTWBYbIUOZzO9n2w3MBK822ob6B3LzcQUuzlQxgZ0r5CKjkW2SI0lohg8G0e0opKCUyNGmt6B8FpXrKYwe
los2Adt8TGeoMw4inukajUcIeJ9ke16AsHpIh4kGpoDKlRIYorRUyGGJBKZcanYsMSVor+kdBqZ6yg1LuiB2U0l1
vERkEjY2NfPTRR/Hvd+7cyYYNGxgxYgTjx49P4cjkqGI9pQz1lBKRgaW1YoiLB6WCKR6IiAxn6bpWKCjVU3b5njv
cBChTSkQGxzvvvMNZZ50V//7GG28E4Morr+Txxx9P0aikRzxt5XsKSonIQNJaMcTFyveiKt8TkYGTrmuFglI9Fc+
UUqNzERk8ixYtwjRV+jUktespFY7o71BEBo7WiiHOYzU6d5sKSonIwEnXtUJFjj1lB6VcdqZUSOV7IiLSHZXviYh
IDxj27ntuZUqJSAZSUKqn7PI9p8r3RESkJxLK99LvrpSIiKQHw22V7ylTSkQykYJSPRXLlArZmVIRMy1T30REJE2
4rTvf6iklIiLdMez1wmOq0bmIZB4FpXrKzpRy2JlSgO58i4hI19x+AHxGkHBUQSkREemc086s9ShTSkQykIJSPWV
nSjmCbUEplfCJiEiX7IuMbJXviYhINwyPVb6nTCkRyUQKSvWUHZQyQo3xQ2p2LiIiXYrvvteq8j0REelSLFPKi4J
SIpJ5FJTqKbt8zwg04jCsQ8qUEhGRLsWCUkaQsDKlRESkC06P1VPKq0wpEclACkr1lJ0pRbAJj8v6YwsqU0pERLp
iN67NVqNzERHphtOrTCkRyVwKSvVUPCjViNtppUrpIkNERLrksRqdZ6mnlIiIdMNp95TyEsTUxhgikmEUlOopu3w
PTPId1l0Mle+JiEiXYplSRoCw1gsREemC286UchomkbCypUQksygo1VPubDCsP64Cl7VdayisO98iMrDuuusuTjr
pJHJzcxk1ahSf/vSn2b59e6qHJT3htjKlfASVWSsiA0prxdAWK98DCLW2pHAkIjKcpetaoaBUTxlGvIQvL54pFUn
liEQkA6xcuZJvfetbrFmzhhUrVhAOh1m6dClNTU2pHpocjSe2+16AUFQ3MURk4GitGNrcniyiptUeJBzU35mIDIx
0XStcKf30ocbjh0A9eU4rUyqoTCkRGWAvvfRSwvePPfYYo0aNYt26dzxxxhkpGpX0iF2+5zYiRIKBFA9GRIYzrRV
Dm9vpJIAbH0HCrc2pHo6IDFPpulYoKNUbsUwpoxVQo3ORIc00IWT/wy8atR4HneAYhARSd7aVfdkHdXV1AIwYMSK
ZI5KBYJfvATjCKscQGZK0VsggcDgMWvHgI0gkpPVCZMjRWtEvCkr1ht3sPNdhBaWCYQWlRIasUDPcWQZYdcwFg/n
Z/3YgvjNbb5imyY033sjpp5/OzJkzB2BgklRON1GcOIhghHXnW2RI0lohgySAB4BwQOuFyJCjtaJfFJTqDTtTKke
ZUiKSAt/+9rfZtGkTq1atsvVQpCcMg5AzC2+kCSOkiwwRGRxaK4amWFAqGlSmlIgMvHRaKxSU6o0jglJBBaVEhi5
3tnVnAYhGo9Q3NJCXm4tjsNJse+n666/nueee4/XXX2fPnj1cddVVXHfddVx//fXcfvvtvPDCC+zevZvf/OY3
oKHBgwfz0EMPsWfPHp5//nmuvfZaLrvsMgCuvPJKXnrpJZ5++mn27NnDQw89xOuvv86VV14JwIUXXsiW
LVt4+umn2bNnDw899BCvvfYaV1xxBQDXXXcdb7zxBk8//TR79uzhoYce4rXXXuOqq64C4Prrr+eNN97g
6aefZs+ePTz00EO89tprXHXVVQBcf/31vPHGGzz99NPs2bOHhx56iNdee42rr74agOuvv54333yTp556
ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpqAK6//nrefPNNnnrqKfbs
2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut58803eeqpp9izZw8P
PfQQr732GldffTUA119/PW+88QZPPfUUe/bs4aGHHuK1117j6quvBuD666/nzTff5KmnnmLPnj08
9NBDvPbaa1x99dUAXH/99bz55ps89dRT7Nmzh4ceeojXXnuNq6++GoDrr7+eN998k6eeeoo9e/bw0EMP
8dprr3H11VcDcP311/PGG2/w1FNPsWfPHh566CFee+01rr76agCuv/563njjDZ566in27NnDQw89xGuv
vcbVV18NwPXXX88bb7zBU089xZ49e3jooYd47bXXuPrqqwG4/vrrefPNN3nqqafYs2cPDz30EK+99hpX
X301ANdffz1vvvkmTz31FHv27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42rr74a
gOuvv54333yTp556ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpqAK6//nre
fPNNnnrqKfbs2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut58803eeqp
p9izZw8PPfQQr732GldffTUA119/PW+88QZPPfUUe/bs4aGHHuK1117j6quvBuD666/nzTff5KmnnmLP
nj089NBDvPbaa1x99dUAXH/99bz55ps89dRT7Nmzh4ceeojXXnuNq6++GoDrr7+eN998k6eeeoo9e/bw
0EMP8dprr3H11VcDcP311/PGG2/w1FNPsWfPHh566CFee+01rr76agCuv/563njjDZ566in27NnDQw89
xGuvvcbVV18NwPXXX88bb7zBU089xZ49e3jooYd47bXXuPrqqwG4/vrrefPNN3nqqafYs2cPDz30EK+9
9hpXX301ANdffz1vvvkmTz31FHv27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42r
r74agOuvv54333yTp556ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpq
AK6//nrefPNNnnrqKfbs2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut5
88032bVrF3v27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42rr74agOuvv54333yTp556
ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpqAK6//nrefPNNnnrqKfbs
2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut58803eeqpp9izZw8P
PfQQr732GldffTUA119/PW+88QZPPfUUe/bs4aGHHuK1117j6quvBuD666/nzTff5KmnnmLPnj08
9NBDvPbaa1x99dUAXH/99bz55ps89dRT7Nmzh4ceeojXXnuNq6++GoDrr7+eN998k6eeeoo9e/bw0EMP
8dprr3H11VcDcP311/PGG2/w1FNPsWfPHh566CFee+01rr76agCuv/563njjDZ566in27NnDQw89xGuv
vcbVV18NwPXXX88bb7zBU089xZ49e3jooYd47bXXuPrqqwG4/vrrefPNN3nqqafYs2cPDz30EK+99hpX
X301ANdffz1vvvkmTz31FHv27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42rr74a
gOuvv54333yTp556ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpqAK6//nre
fPNNnnrqKfbs2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut58803eeqp
p9izZw8PPfQQr732GldffTUA119/PW+88QZPPfUUe/bs4aGHHuK1117j6quvBuD666/nzTff5KmnnmLP
nj089NBDvPbaa1x99dUAXH/99bz55ps89dRT7Nmzh4ceeojXXnuNq6++GoDrr7+eN998k6eeeoo9e/bw
0EMP8dprr3H11VcDcP311/PGG2/w1FNPsWfPHh566CFee+01rr76agCuv/563njjDZ566in27NnDQw89
xGuvvcbVV18NwPXXX88bb7zBU089xZ49e3jooYd47bXXuPrqqwG4/vrrefPNN3nqqafYs2cPDz30EK+9
9hpXX301ANdffz1vvvkmTz31FHv27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42r
r74agOuvv54333yTp556ij179vDQQw/x2muvcfXVVwNw/fXX88Ybb/DUU0+xZ88eHnroIV577TWuvvpq
AK6//nrefPNNnnrqKfbs2cNDDz3Ea6+9xtVXXw3A9ddfz5tvvslTTz3Fnj17eOihh3jttde4+uqrAbj++ut5
88032bVrF3v27OGhhx7itdde4+qrrwbg+uuv580332TXrl3s2bOHhx56iNdee42rr74agOuvv543

B5/MlLCpDUXZ2NuPHjx+yi+BwY9iNzr3KlBJJe1orJJWiTitTyqFMKZG0prUi+RSU6g37jreDKD4CCkqJDCEej4f
x48cTDoeJRCIJz4VCIV5//XXOOOMM3G53ikbYf06nE5fLNeQXwOHE8FpBKY8ZSPFIRKQntFZIqkTtTClHROuFSLr
TWpFcCkr1hscPGIBJDq0q3xMZYgzDwO12d1ggnE4n4XCYrKysIb14SPox7EbnWcqUEhkytFZIKsQypQwFpUSGBK0
VyaOc3d4wjHgJX7bRSlCZUiIi0g2nnSmVpUwpERHpRtRlZUo51VNKRDKMglK95bWCUjm0qnxPRES65fBamVJeFJQ
SEZGumXb5njOq9UJEMouCUr1l95Xy06LyPRER6VYsKOUjgGmaKR6NiIikK9POlHKpfE9EMoyCUr1ll+/5jVZCEV1
giIhI11ztglKRqNYMERHpnOmOZUqpfE9EMouCUr3lzQXAr0bnIiJyFM4s60aGjwBhBaVERKQLhssHgEvleyKSYRS
U6q2ETCkFpUREpGsuu9G51wgTCgVTPBoREUlXph2UcisoJSIZRkGp3rJ7SuXQot33RESkW247Uwog1NqcwpGIiEg
6MzxW+Z7bDIJ6EIpIBlFQqrfs3fdUviciIkfjcGcRMQ0Aoq2NKR6NiIikK9PddhODQEPqBiIiMsgUlOotu3wvW+V
7IiJyNIZBK14AQoGmFA9GRETSlcOTTbNprRc0H07tYEREBpGCUr1lNzrPoUW774mIyFG12EGpqIJSIiLSBZfTQTX
WdQZNCkqJSOZQUKq37J5SfkPleyIicnSthoJSIiLSPY/TwWEzz/pGQSkRySAKSvWWXb6XQyvBSBRTjQhFRKQbrVj
Na6NBNToXEZHOuZwG1bGglMr3RCSDKCjVW3b5XjatACrhExGRbgXsTCkzqEwpERHpnNvpoJpYptSh1A5GRGQQKSj
VW3amlN9oAVCzcxER6VbAUKaUiIh0z+002pXvVaV2MCIig0hBqd6ye0rlxDOlFJQSEZGuxYJSypQSEZGuuJ0Oqk2
70bnK90Qkgygo1VveWKaUFZRSs3MREelO0GFv8R1qSe1AREQkbblUviciGUpBqd7yWHcw/HamVFCZUiIi0o2gw2c
/UPmeiIh0LrF8T5lSIpI5FJTqrVimFC2AqUwpERHpViieKaWglIiIdM4q34vtvqeeUiKSORSU6i27p5TTMMkiqN3
3RESkWyE7U8pQUEpERLrgdjqoMtuV75m6xhCRzKCgVG+5/fGHObSq0bmIiHQr5LQanRthBaVERKRzLodBVaynVCQ
IgYbUDkhEZJAoKNVbDgd4rBK+bKOVgMr3RESkG+F4ppQanYuISOc8LgeteGnBLvnWDnwikiH6FZS66667MAyDG26
4IX7MNE2WLVtGWVkZPp+PRYsWsXXr1oTXBQIBrr/+eoqLi/H7/Vx88cXs27evP0MZXHZQKocWZUqJiBxFxq4VtrD
TCko5wwpKiYh0JdPXCpfDAGjLllKzcxHJEH0OSq1du5b//u//Zvbs2QnH77nnHu677z4eeugh1q5dS2lpKUuWLKG
hoS0F9YYbbuDZZ5/l6aefZtWqVTQ2NnLhhRcSiUT6PpPBZPeV8qt8T0SkWxm9VtgiLqt8zxFRUEpEpDNaK6yeUkB
bs3MFpUQkQ/QpKNXY2MiXvvQlfvnLX1JYWBg/bpomDzzwAN/73ve45JJLmDlzJk888QTNzc089dRTANTNTV1fHoo49
y7733snjxYubNm8eTTz7J5s2befnll5Mzq4EW24HPaNHueyIiXcj4tcIWUaaUiEiXtFZYYkGpKjPXOtB0KIWjERE
ZPH0KSn3rW9/iggsuYPHixQnHd+7cSUVFBUuXLo0f83q9nHnmmaxevRqAdevWEQqFEs4pKytj5syZ8XPSnsdaLNT
oXESkaxm/VtjayvfU6FxE5EhaKyxup1W+dzhqZ0qpp5SIZAhXb1/w9NNPs379etauXdvhuYqKCgBKSkoSjpeUlLB
79+74OR6PJ+FOSOyc2OuPFAgECAQC8e/r6+sBCIVChEKh3k6h35zubBxYjc6bA70bQ+zcVIx7sGTCHEHzHE4yYY4
wuPPTWtEmau++54y09ngcmfAzmQlzBM1zOMmEOYLWilQxTOtGd6ynVKShkuhRxpMJP5OZMEfQPIeTTJgjJHd+vQp
K7d27l3/+539m+fLlZGVldXmeYRgJ35um2eHYkbo756677uL222/vcHz58uVkZ2f3YOTJNf9wPWOxMqXWrd+Ac9+
7vX6PFStWJH9gaSYT5gia53Ay3OfY3Dw4mTpaKxLtO1RnPQg08OKLL/bqtcP9ZxIyY46geQ4nw32OWitSI2oCuKi
2y/cOfLiJ9aGerRnD/WcSMmOOoHkOJ8N9jslcK3oVlFq3bh2VlZXMnz8/fiwSifD666/z0EMPsX37dtbWsesp1VINUa0bIiLSUax8rwq7fK/pEKh
4OZWVlfG7HKWlpQSDQWpqahLualRWVrJw4cJOP/e2227jxhtvjH9fX1/PuHHjWLp0KXl5eb2ZQlI4/7ICNqzBTwu
TZ87i/BPH9vi1oVCIFStWsGTJEtxu9wCOMnUyYY6geQ4nmTBHgKqqqkH5HK0Vifa2/BXehyxHmPPPP79Hr8mEn8l
MmCNonsNJJswRtFak0vfW/52qcD4AYwq9lB5lzciEn8lMmCNonsNJJswRkrtW9Coodc4557B58+aEY1dffTXHHXc
c3/3ud5k8eTKlpaWwaWsLGCefPmARAMBlm5ciV33303APPnz8ftmZyXIyY46geQ4nw32OWitSI2oCuKi
djrvd7tT8RfusxcJvtBIxjT6NIWVjH0SZMEfQPIeT4T7HwZqb1ooj2JtjuCOtvR5Hysc+CDJhjqB5DifDfY5aK1L
H73VRFbYypRzNVTh6OJ50GPtAy4Q5guY5nAz3OSZzbr0KSuXm5jJz5syEY36/n6KiovjxG264gTvvvJ
TZ87i/BPH9vi1oVCIFStWsGTJEtxu9wCOMnUyYY6geQ4nmTBHgKqqqkH5HK0Vifa2/BXehyxHmPPPP79Hr8mEn8l
MmCNonsNJJswRtFak0vfW/52qcD4AYwq9lB5lzciEn8lMmCNonsNJJswRkrtW9Coodc4557B58+aEY1dffTXHHXc
c3/3ud5k8eTKlpaWwaWsLGCefPmARAMBlm5ciV33303APPnz8ftmZyXIyY46geQ4nw32OWitSI2oCuKi
djrvd7tT8RfusxcJvtBIxjT6NIWVjH0SZMEfQPIeT4T7HwZqb1ooj2JtjuCOtvR5Hysc+CDJhjqB5DifDfY5aK1L
H73VRFbYypRzNVTh6OJ50GPtAy4Q5guY5nAz3OSZzbr0KSuXm5jJz5syEY36/n6KiovjxG264gTvvvJ
U7rzzTrKzs7n88ssBuOSSS7jzzjvJzs7m8ssvB+CSSy7hzjvvJDs7m8suuwyASy65hDvvvJPs7GwuvfRSAC65
5BLuvPNOsrOzufTSSwG45JJLuPPOO8nOzubSSy8F4JJLLuHOO+8kOzubSy+9FIBLLrmEO++8k+zsbC699FIA
LrnkEu68806ys7O59NJLAbjkkku48847yc7O5rLLLgPgkksu4c477yQ7O5vLLrsMgEsuuYQ777yT7Oxsrrjs
MgAuueQS7rzzTrKzs7n88ssBuOSSS7jzzjvJzs7m8ssvB+CSSy7hzjvvJDs7m8suvRSAyy65hDvvvJPs7GwuvfRS
AC65

ViIgMYwpK9ZHT7inlN1oJKlNKRES64fe6qMHefc+MQmttSscjIiLpKdbo/HBsx9ZoCAL1KRyRiMjAUlCqjwy7p1Q
OLYQiunshIiJdy/G6COMi5I6VY6ivlIiIdBTLlKoLu8Dttw5qzRCRYUxBqb6yM6WyCRAMRVI8GBERSWd+u3FtwFN
gHVBfKRER6UR8971ABPzF1kEFpURkGFNQqq88VlDKbUQIB1tSPBgREUlnfrtxbYu70DqgLb5FRKQTsUbnTYFwW1B
Ka4aIDGMKSvWVxx9/GG5pSOFAREQk3cXufDe77Ma1ypQSEZFOtGVKhSFbmVIiMvwpKNVXDicteAGItCooJSIiXYt
dZDQ4C6wDusAQEZFOxNYLK1NqpHWw6VAKRyQiMrAUlOqHFsMHQDSgoJSIiHQttptSvWE3Om+uTuFoREQkXeUkBKW
KrIPKrhWRYUxBqX6IBaXM1sYUj0RERNJZrNF5bTwopUwpERHpKLZeNAYiKt8TkYygoFQ/BGJBqaAypUREpGuxRuf
VZq51QHe9RUSkEzkq3xORDKOgVD+0OrIBMAPKlBIRka7FLjKqonZQSne9RUSkE/HdWkMRIr5Y+Z7WDBEZvhSU6oe
AHZQi2JTagYiISFqLNa6tjORYB9RTSkREOhFbLwBavIXWgyZl14rI8KWgVD8EnFZQylCmlIiIdCOWKVURD0rprre
IiHTkdTlwOQwAmuI7th4C00zdoEREBpCCUv0Qclg9pRwhBaVERKRrsTvfFSG/dSDUDMHmFI5IRETSkWEY8TWj0Zl
vHYyGIFCfwlGJiAwcBaX6IeiyMqUUlBIRke7EeoQcCrjB6bEOqtm5iIh0IpZd2xDxgMfOsFUvQhEZphSU6oeQ07r
jbYR0t1tERLoW2+K7KRiBbDWuFRGRrsVuZDQFwm1rhoJSIjJMKSjVD2GXFZRyKlNKRES6ESvFaApGMONBKWVKiYh
IR/HyvUAY/MXWQd3IEJFhSkGpfojY5XuusDKlRESkazntdlOK+EZYD7SbkoiIdCK2ZjQFwuAfaR1sOpTCEYmIDBw
FpfohYmdKucJNKR6JiIiksyy3A3szJcJeOyilTCkREelEYsm3nSml8j0RGaYUlOqHiN14UJlSIiLSnfa7KQU8hdZ
BlWKIiEgnstv3lPKr5FtEhjcFpfoh6rYypTwRZUqJiEj3YuUYLe4C64AuMEREpBMq3xORTKKgVD/EglLuiDKlRES
ke7FMqWZXgXVApRgiItKJhEbnKt8TkWFOQan+sMv3vFEFpUREpHvxiwxnvnWguTqFoxERkXSVkCmlHVtFZJhTUKo
fTDso5Ym0pHgkIiKS7nLsHiH1RiwopbveIiLSkd8T6ykVgSx7zQjUp3BEIiIDR0GpfjCycgFwEYaQAlMiItK12G5
KtYa1duiut4iIdCahfC8rzzrYqqCUiAxPCkr1hyeXoGndyVCdt4iIdCdWjlGDfYHRXA3RSApHJCIi6SihfM9rrxm
BejDNFI5KRGRgKCjVD26Xk+r4xYWCUiIi0rXYFt9VUTtTChNaalM2HhERSU+JmVJ2+V40DCH1sRWR4UdBqX7wuBx
Um3ZQqkllGCIi0rXYRUZDiLaLDN3QEBGRI8TWi6ZgGDx+MOzKDJXwicgwpKBUP3icBlVmrDeILixERKRrOR7tpiQ
iIkfXVr4XAcMAr329oWbnIjIMKSjVDx6Xg6pY+Z56SomISDcSyjGyi62DWjtEROQIfrvcuzEQtg6o2bmIDGMKSvW
D29m+fO9QagcjIiJpLaFxrTKlRESkC7H1IhiOEopE20q+W+tSOCoRkYGhoFQ/uJ0Oqkw1OhcRkaPzty/H8MeCUlo
7REQkUWy9gNgOfHZQKqCglIgMPwpK9YPH5aAau8Zbjc5FRKQbCeUY8Uyp6hSOSERE0pHb6cDjsi7TrB34VL4nIsO
XglL94Glfvqe73SIi0o2c9rspqaeUiIh0I6HZude+3lCjcxEZhhSU6gerfC+WKaULCxER6ZpfPaVERKSHErJrlSk
lIsOYglL9YJXvxTKldGEhIiJdy2m/+57fzpRSlq2IiHTC72l3I0ONzkVkGFNQqh/cToPDZrt02nAgtQMSEZG0Fcu
Uag1FCWcVWgfVU0pERDqRsGOryvdEZBhTUKofPC4H9fgJmVZ6rbKlRESkK7FSDIAWjx2UUum3iIh0wt8+u1bleyI
yjCko1Q8epwMwqInvwHcopeMREZH05XE6cDkMAJocdilGuAWCzSkclYiIpCNlSolIplBQqh9idzDU7FxERI7GMIy
2O99mFjg91hPqKyUiIkeIZdc2BSPqKSUiw5qCUv3gdjrIcjuoNtXsXEREji7e7DwY1Q58IiLSpexOG50rU0pEhh8
FpfopN8tNNcqUEhGRo4vf+Q6EIdvega9JQSkREUnUefmeMqVEZPhRUKqfcrNcVMUzpRSUEhGRriU0rs0eYR1UppS
IiByhbb2ItDU6DzSAaaZwVCIiyaegVD/lZrnbyveUKSUiIt1IuPPttzOldENDRESOkNM+szaWKWVGIdiYwlGJiCS
fglL9lJflogr1lBIRkaPzt+8Rop5SIiLShVimVFMwDG4fONzWE2p2LiLDjIJS/ZRQvqdMKRER6UZCOUa8p5TWDhE
RSZRQ7m0YbSV8anYuIsOMglL9lJflptqMNTo/lNrBiIhIWksox1BPKRER6UJCuTe0a3auoJSIDC8KSvVTbkL5nu5
2i4hI1xLufMd7SikoJSIiieLle4GIdUCZUiIyTPUqKPXzn/+c2bNnk5eXR15eHqeeeip//etf48+bpsmyZcsoKyv
D5/OxaNEitm7dmvAegUCA66+/nuLiYvx+PxdffDH79u1LzmxSILd9plRrHURCqR2QiEiKaa3omr/9nW9fLFOqOoU
jEhFJDa0V3Ytl1jYqU0pEhrleBaXGjh3LT37yE9555x3eeecdzj77bD71qU/FF4h77rmH++67j4ceeoi1a9dS
KkiVLaGhoiL/HDTfcwLPPPsvTTz/NqlWraGxs5MILLyQSiR3ZoMkN8tFLTlEY3+UuuMtIhlOa0XX/B7rIqM5GGk
r32tRUEpEMo/Wiu61v4lhmiZk5VtPtNamblAiIgOgV0Gpiy66iPPPP59jjz2WY489lpsmyZcsoKyvXs3ABUVFXg
8HgoLCzucU1FR0eVnBgIBAoFA/Pv6euuXcSgUIhRKbQ+nbJcBQDV5jAfC9QcxuxlTbLypHvdAyoQ5guY5nGTCHGF
w56e1onN2TIrG1hAhXLhcPoxwC6GGQ+DMjp+XCT+TmTBH0DyHk0yYI2itSCc+t4OWUJTaplYK3H6cQKS5lugRP4v
pNu5kyoQ5guY5nGTCHCG58+t1UGratGls2LCB2tpa/vCHP3DllVeycuXK+POGYSScb5pmh2NHOto5d911F7fffnu
H48uXLyc7O7uTVwyevY0ALg6G/OCAbe+8zs7dWUd93YoVKwZ8bKmWCXMEzXM4Ge5zbG5uHrTP0lrRuZ0NAC4O1Tb
w4osvstTIwkcLq19+ntrsyR3OH+4/k5AZcwTNczgZ7nPUWpE+XDgBg7/9/TU+0XSAWUD5zvdY9+KLCecN959JyIw
5guY5nAz3OSZzreh1UMrj8XDMMccAcOKJJ7J27Vr+67/+i+9+97t+97uAdddi9OjR8fMrKyvdzlKS0sJBoPU1NQk3NW
orKxk4cKFXX7mbbfdxo033hj/vr6+nnHjxrF+umEhjxrF06VLy8vJ6O4Wk2l3dzH9uXkUVVp33jAklHL/o/C7PD4VCrFixgiV
LluB2uwdrmIMqE+YImudwkglzBKiqGryNGLRWdO6Dgw08sOVNok4P559/Fq79d0NlDafNm4455ez4eZnwM5kJcwT
NczjJhDmC1op0ct/2VTRUNzPv5FM5vqYG9j9F2YgcSs63rjUy4WcyE+YImudwkglzhOSuFb0OSh3JNE0CgQCTJk2
itLSUFStWMG/ePACCwSArV67k7k7rvvBmD+/Pm43W5WrFjBZZddBkB5eTlbtmzhnnvvvfPm43aabbqKoqIgRI0Zw
ROT4AKqM54ABnazXOHowpHcY+0DJhjqB5DifDfY6pnFumrxUx+X4rk7YpGLHGZPeVcgXroZMxptPYB0omzBE0z+F
kuM9Ra0X6yLF717ZGwOW3Am+OYCOOI8aZjmNPtkyYI2iew8lwn2My59aroNS//du/cd555zFu3Dh3zFu3Dga
113jppZcwDIMbbriBO++8k6lTpzJ+/HiuvPJKbrvtNiZMmMDUqVPZsGEDAHfddRdTpkxh/PjxXHnllcyfP5+
0SQ2mWKPzKlONzkVEQGtFd2KNzoPhKKFIFHd2ux34REQyiNaKo/O33xzDr0bnIjI89SoodfDgQb785S/z

+s2fP5qWXXmLJkiUA3HrrrbS0tHDddddRU1PDggULWL58Obm5ufH3uP/++3G5XFx22WW0tLRwzjnn8Pjjj+N0OpM
7s0HidjrwuZ1UR+yFomnwUp5FRNKR1oquxS4wwLrIKPDZJSct1SkakYhIamitOLqEHVuL7GuNVgWlRGR46VVQ6tF
HH+32ecMwWLZsGcuWLevynKysLB588EEefPDB3nx0WsvNclHdZC+QypQSkQyntaJrbqcDj8tBMBylMRCmwGdnSjU
rKCUimUVrxdHFbmQ0BiKQVWAdbK1L3YBERAaAI9UDGA5ys1xt5XtNCkqJiEjX2u58R+I9pZQpJSIiR8rxWhlfTYE
weO1rjWADRCMpHJWISHIpKJUEuVnutqBUS40WChER6VK2x77ICIbBp55SIiLSuWxPu/K9rHY7AwYaUjQiEZHkU1A
qCXKzXNSQi4kBmCrDEBGRLiX0CMlW+Z6IiHQu3ug8GAaXF5z2roFqdi4iw4iCUkmQl+UmioOAO9860HQotQMSEZG
0lbCbkhqdi4hIF9rK9+wqjCz7WkPNzkVkGFFQKglys6wLjGZXgXVAzc5FRKQLCY1r443OVb4nIiKJ2taLsHUgVsK
nZuciMowoKJUEsaBUYywopWbnIiLShYTGtbHyvUAdRMIpHJWIiKSbhHJvaGt2rvI9ERlGFJRKgtwsNwD1hp1S21y
VwtGIiEg683va3fmObfEN0FqbkvGIiEh68nuOCErFM6UUlBKR4UNBqSSIZUrVYC8UypQSEZEuJPSUcrrAG7uhob5
SIiLSpkP5njKlRGQYUlAqCfLsTKkqcq0D6iklIiJd6FCOka1m5yIi0lHbenFko/Pa1AxIRGQAKCiVBLFMqUNRZUq
JiEj3EhqdQ7tm5wpKiYhIG3/7HoSg3fdEZFhSUCoJYj2lDob91gEFpUREpAs5R15kxJqdt2gHPhERaRPPlAqGMU1
T5XsiMiwpKJUEsUyp8lCOdUDleyIi0gV/u4sMAHwq3xMRkY5i60XUhJZQRI3ORWRYUlAqCWI9pfYFlSklIiLd69C
4VuV7IiLSiWyPE8OwHls7ttrleynKlKptDvLnDfsJR6Ip+XwRGZ5cqR7AcBDLlDoQ8oMT6253NAoOxfxERCRRbIv
v5lhPqXj5noJSIiLSxjAM/B4XjYGw1ew8Vr7XWpeS8dz2x838dUsFwXCUz504LiVjEJHhR1GTJMixg1I1sd33zKh
6g4iISKdijWs7ZEpp3RARkSMkNDtPYfleQ2uIv79fCcB75Q2D/vkiMnwpKJUEbqcDn9tJGBdRr51Wq75SIiLSiZw
je0plq3xPREQ6l1DyncJG539/r5Jg2Crb213VNOifLyLDl4JSSRIr4QtlFVkH1FdKREQ6EW90Hs+UKrC+KlNKRES
OkNN+zUhhptRfNpfHH++ubh70zxeR4UtBqSSJBaUCHnsXJWVKiYhIJ2JBqVDEJBCOqNG5iIh0KdaH0Gp0XmAdDDV
BJDRoY2hoDbHyg0Px7/dUNxONmoP2+SIyvCkolSR5PmsHvha3HZRqOtTN2SIikqn8Hmf8cVMgokbnIiLSpbbs2gh
4c9ueCAxeX6dX3rdK9yYWZeNyGATDUQ42tA7a54vI8KagVJLkZllBqSZXgXWgqSp1gxERkbTlcjrIclvLb1Mg3JY
pFW6FUEsKRyYiIukmp32jc6cb3NnWE4O4A99fNlmlexfNKWNsoQ+A3VUq4ROR5FBQKkli5XsNTjU6FxGR7uUkNK7
NBYf1vUr4RESkvYRG5zDozc4bA2Fes0v3zp81mvFFfkDNzkUkeRSUSpI8OyhVZ9hBKTU6FxGRLiQ0OzcM8Nml3yr
hExGRdnKO3BxjkJud//29gwTDUSYX+zmuNJcJI6xMLWVKiUiyKCiVJLHyvWrshUKZUiIi0oWExrWgZuciItKpbHu
9aArGglL2DfBBKt970d517/xZozEMgwlFdlBKO/CJSJIoKJUkufZdjKqo3YBQPaVERKQLOe0b10K7Zuc1KRqRiIi
kI3+8p5S9Xgxi+V5jIMyr29tK9wDG25lSe5QpJSJJoqBUksR6SlVGlSklIiLdy45dZASPyJRS+Z6IiLSTyvK9WOn
epGI/x4+2brxPLLZ6Su2qasI0zQEfg4gMfwpKJUmsfK88ZP2iprkKotEUjkhERNKV/8iLjFhPKZXviYhIO6lsdN5
WuleKYRhAW6ZUQ2uY2ubQgI9BRIY/BaWSJJYpdSAWlIqGobU2dQMSEZG0leM5IiiVHWt0rvI9ERFpk+ezbnzXtdg
BoEHqKdUUCPPaEaV7AFluJyV5XkB9pUQkORSUSpJYplRNgLY7GM3qKyUiIh213fm2e4T41FNKREQ6GpVrBYAqGwL
WgXj53sAGpf7+fiWBcJSJRdlMH52X8NyEEdZN+N1VTQM6BhHJDApKJUksU6q+NQzZRdbBJvWVEhGRjnLijWtjmVL
afU9ERDoqzcsCoLopSCAcAa+dKTXA5Xsvbkrcda+98UVqdi4iyaOgVJLk26m1Da0h8BdbB9XsXEREOtFlTyk1Ohc
RkXYKst14XNYlW2V9YFAanTcFwry6vRJILN2LmWgHpXYpKCUiSaCgVJLEMqVaQ1GiPmVKiYhI1zo0rvUpU0pERDo
yDCPew+lgfeugNDp/bfuheOnejLK8Ds+PL7LK9/ZUq3xPRPpPQakkiW3XChDKil1cKCglIiIdxbf4Dh5RvqdMKRE
ROUKshO9gfWBQGp2v32P1N1w0bVSH0j2ACfYOfLuVKSUiSeA6+inSEy6ng2yPk+ZghFbPCLwATWp0LiIiHXXb6Dw
aTdGoREQkHY2yg1IV9a0waoDK96JROPQ+7H6DJVv+zGWenRw2bgJmdDh1gl2+V9kQoDkYJtujS0oR6Tv9Bkmi3Cw
XzcEIza5C8kGZUiIi0in/kY3OYz2lzKhVkuHyp2hkIiKSbmKZUpXJLt8zTdj4G3j/L7B7dTxb9xQAB4zf/b/AVzu
8rCDbQ16Wi/rWMHuqmzmutGOJn4hIT6l8L4lys6xm540uO6226VAKRyMiIukq58hG5+4scFt3nlXCJyIi7cV6SlX
Ut7Y1Og+3QiTYvzeufA/+9E14/wVr7XFnExh/Bj8NfxoA3+HN0HCw05dOLLZunqiET0T6S0GpJIo1O69zFFgH1Oh
cREQ6ESt1iDc6h3bNzmtSMCIREUlXJfGeUu0ypaD/JXw1u6yvIybDNS/Dv+5h1an/w33hy/jAeYz13Ecvd/rS8XZ
fqT0KSolIPykolUSxTKkawy7DaKxM4WhERCRdxTKlmoMRTNO0Dmbba0eLglIiItKmpH2jc4cTPLnWE4F+NjtvKLe
+jjwexp0ETjfbDliBrp2FC63nPlze6UtjfaV2awc+EeknBaWSKJYpdZgC60DTIYiEu36BiIhkpFhPqUjUJBC2G5v
7tAOfiIh0VNouU8o0zXgJn9HfTKmGCutrbmn80HsV1nu2TDjbOvDxqxAJdXjphBEq3xOR5FBQKonyYkGpaA4YDsB
Us3MREenA326nosYjm503KyglIiJtYplSzcEIDYFwu2bnDf1748ZYUGp0/FAsU6r42IXWzZJAHex9u8NLx8cypRS
UEpF+UlAqiWLle3WtUfCPtA42dt4cUEREMpfDYZDtOWIHvmxlSomISEc+jzN+8/tgXbtm5/3dge+ITKnGQJjd1Va
Q6fgxBXDMYuv5Tkr4JhZZmVL7a1sIRaL9G4eIZDQFpZIotlg0tIYhp8Q6qL5SIiLSCb/3iGbn8fI99ZQSEZFECX2
lsuydvluT1FPKzpTaXlGPaVq7/RXleGHqUuv5D1d0eOmoXC9el4NI1ORAbUv/xiEiGU1BqSSKZUo1BEJtQanYHQg
REZF2Ys3OmwIR60AsU0rleyIicoTSfCsoVdFuBz4jaZlS1nXLtnKrHPD40XYm1jHnAAZUboW6/QkvdTiM+A58KuE
Tkf5QUCqJcjvNlFL5noiIdBRrdt7UIVNKQSkREUk0Kret2XlSyvciIWtTJohnSsX6SU2PBaWyR8DYk6zHH3XMlor
vwFelHfhEpO8UlEqiWKZUfWsYckZZB1W+JyIinYg1O1ejcxEROZrSfC9gB6W8SQhKxW6cO9zxmyLvlVvvF8+Ugm5
L+MZrBz4RSQIFpZKoLVMq1La1qjKlRESkE23le2p0LiIi3SvNa58pZfWUMlr7sfte+ybnDqs31PsVdqZUWfug1BL
r647XIBxIeIuJxXamVLWCUiLSdwpKJVFi+Z4ypUREpGtdNzqvTc2AREQkbY3Ki/WUCrQr3+tHo/N4k3PrRvquqiZ
aQ1F8bmd8Zz0ASmdbbUmCjbDnzYS3iPWU2qNMKRHpBwWlkigv1ui8tV2j80Y1OhcRkY78XTU6D9RbvT5ERERssUy
pyvpW8Nq77/WnfK99phRt/aSmlebidBht5zkccIydLXVECd8EO3i1u7oJ0zT7PhYRyWgKSiVRLFOqNRQl5BtpHVS
mlIiIdMLvsRqdNwftTKmsfMC+EGipSc2gREQkLZXEglINASKeXOtgazKCUnaT8/JOSvdipi62vn64POHwmAIfDsO
69qlsCHR8nYhIDygolUSx/iAADe4i60GwEQKNKRqRiIikqw7lew5nvE+IglIiItJecY4HhwGRqEm96QPASGKmVKd
NzmMmnwWGEw5/ANU744c9LgdjCq2xqNm5iPSVglJJ5HI6yLbvfDdEveC267GblC0lIiKJOjQ6h3gJn9GqoJSIiLR

xOR0U51g78B0KWYEgAv1pdG73lMpJLN+b3llQylcA40+xHn/0csJTE+I78DX1fSwiktEUlEqyWF+p+pZ2zc4btAO
fiIgkasuUirQdjDU7b9YOfCIikqg0396BL+ixDrTWQV97ObXLlDrcGKCyIYBhwHGluZ2fP7XzvlLji+xm59qBT0T
6SEGpJGvbga99s3MFpUREJJHfa2XWdpYppfI9ERE50qhcKyh1oNW6CW5EQzjMPm6MEd99b3S8dG9ikT9+w6SDqUu
trztfh1BL/PAEewe+XSrfE5E+UlAqyWJBqfrWdplSanYuIiJHiGfWtra7oPAVAmC0KFNKREQSleZb5Xv7mxzENsZ
wR/oQDAoHILbO5Ja26yfVRZYUwKjpkDcGwi2wa1X88IRYppTK90SkjxSUSrJc+yKjoTUUbxyoTCkRETnSqDzr4uJ
gfWvbQV+SMqXCAXj6S/D3H0I02r/3EhGRtFBiZ0pVNAQhy+r95I60dPeSzsVK95xe8BV2308qxjBgytnW413/iB+
eUGT3lFL5noj0kYJSSdZWvtc+U0pBKRERSTQ632pUe7gxSCBs95WKNTrvb1Bq71vw/gvwj3vhhX9WYEpEZBgosXt
KVdQHwGvt1tqnTKn2O+8ZBu+VWw3TO915r70JC62ve9bED423y/dqm0PUtfSxlFBEMpqCUknWlikVVk8pERHpUmG
2G6/LWoYr6uxsKbt8r9+ZUjW72x6v/xU8/x0FpkREhriSPCsoVVnfGs+UcvUpKNXWT6o1FOGjQ40ATC87SlAqtgP
f/vXxvlJ+ryu+K+CuwyrhE5HeU1AqyfLU6FxERHrAMAzKCqxsqQO1Rwal+tlTqmaX9bV4GhgOePf/FJgSERniSvN
imVKt4O1H+V7s2iS3lI8qG4lETQqy3fH371LhJOv6JhqCA+/GD08ZaZXw7Tjc2PuxiEjGU1AqyRLL92JBKTU6FxG
RjkbbpRjldfZFRbLK92rtTKm5X4RLftkuMHV9p4GplmCENz46jNnXrcVFRGTAxYJGtc0hIh6rKXn/MqVKE/pJGYb
R/esMoy1bas+b8cNTRuUA8FGlglIi0nsKSiVZvHwvEEoMSunutIiIHCHWV6o8Xr6XpEbnsfK9wokw69J2gakn4bm
Ogal/eWYDX/qft/jzhgP9+1wRERkweT5XvOy71WkFgvrV6Dy3lG3xnfeOUroXM/5U62u7vlJTRlpj+bhS5Xsi0ns
KSiVZQqaUvxgwwIxAc1VqByYiImmnrMC6632gNjFTipYa6E/WUixTqmCC9bV9YGrDk/DG/fFTt+yv46Wt1gXKGx8
d7vtniojIgDIMg1I7w7bRsBqMuyN9CAS16ykVC0p1u/Nee/FMqbfiNziOiWVKHVKmlIj0noJSSRbLlKpvDYPTDdl
F1hPqKyUiIkeIZUpVHJEpZUQCOKPBvr1pqKVtzSmc2HZ81qVw7p3W4+1/jR/+r79/GH+8aV9d3z5TREQGRUmuXcK
HVb7nDffs9/bPXv2IL/73Gm753UaqKvYA8HFrDu/FglJHa3IeH8AscPshUAeH3gPaekrtrmoiFFF1iIj0joJSSRZ
vdB7bEjW31PqqoJSIiBxhdCxTKhaU8vjBYd3c8EQa+vamtXvs98pta5wec8wS62vFFoiE2XqgjhXbDhJrI/JhZQP
NwXDfPldERAZciZ0ptdc9CYCC5l1HfU1LMMJ/Lt/Omzuq+N26fbiarOzYr//pAA2tYdxOI16Cd1ROF4w7yXpsl/C
V5fvwuZ2EIiZ7qvvQ40pEMpqCUkmWkCkFkDPK+qpm5yIicoQOjc4NI17C5w73sTdHbOe9wglwZNPaEZPBkwPhFqj
6kJ/aWVIXzi6jNC+LqAlb9tf37XNFRGTAleR6AXjPOAaAvJZ9EG7t9jU7DzdhmpDrdXHr2ePJN6zAUdRv9b89b+Z
oPK5eXBaOi5XwWUEph8Ngsp0t9bGanYtILykolWRtPaXsTKl4s3NlSomISKJY+V5tc4iWYMQ6aJfweSJ9/Id9+yb
nR3I4oHQ2APvfW8PftlpZUt85+xhmj80HYNO+2r59roiIDLhYT6kPWwsw/SNxEME4uKXb1+w4bK0nU0tyuO5EK3i
EO5tXv3cxH9xxHj/94rzeDWJ8YlAK1FdKRPpOQakky7MzpQLhKMFwtF2mlIJSIiKSKC/Lhd/jBOBAXWKzc0+4j/+
wP7LJ+ZFGzwHg/fX/AOCCWaOZWpLLnHEFAGxUXykRkbQ1Ks8KSlU0BDBLrd/nxoF3u31NbFe8KSNz2nbeyykBw+h
dhlTM2BPBcELdHqjb1/beaAc+Eek9BaWSLMfOlAI7W0qZUiIi0gXDMBhdYGVLldfGmp1bfaD6HJRqX77XGTsolVO
z1cqSOmcqgDKlRESGgFI7KFVZ34pZZmU4GeUbun3Nx3b20pRR7YJSuaP7PghvLpTOsh7b2VLKlBKRvlJQKsmcDiN
+17uhNdwuKKWeUiIi0lGsr1Q8UyoWlOph+d6fN+zn0VU7MU3TOtDDTKnpxm4umFHCsSXWDk6zxxQAsLuqmdrmPu7
8JyIiA6okz+opVVHfSnT0XACM8u4zpWLle5OL/e2CUqX9G8j4U62vdlAqlim1o7KxbT0SEekBBaUGQKzZeWJQSpl
SIiLSUVn+EZlS/pEAFDe8B2b3W2s3tIa46bcb+dEL23h1eyWYZrueUp0HpbZHRtNiesg1WviXE93x4/nZbiYWZQO
wSSV8IiJpqcTOlGoNRWkYYfUI5PCHEOh8x9Zo1Gwr3xuVAw3l1hP9yZSCDn2lJhZn4zCgIRCmsiHQv/cWkYyioNQ
ASGh2HgtKNSgoJSIiHY0usPuD1NuZUrM/j+nKYmTjNhxvPtjta9/ZVUM4at2Rvnf5B0SbayBg757XRabUT1/byXv
meACmhD5MeG722AJAJXwiIukqy+0k32fdUKgI59LsLsLAhAMbOj2/or6VllAEl8Ng/IjsJGZK2UGpg1ugtQ6vy2m
9P9qBT0R6R0GpARALStW3htsanQfqINSSwlGJiEg6imVKHYhlSo06jsi5PwHA8dqPYdcbXb52zY6q+OOtB+p5c90
66xv/KPBkdzh/e0UDL24uZ0t0knWgfGPC87G+Ump2LiKSvkrjzc5bqfVPtg4eWN/puTsOWVlS44uycTsdycuUyi2
FwkmACXvXAm19pT5WXykR6QUFpQZAW/leCLLywWUtHOorJSIiR4pt711e13bjwpzzJfYWnoZhRuH3X4XGQ52+Nha
Ummb3hfr7m9aFQWele+FIlFv/sAnTJN5X6sigVGwHPmVKiYikr5L8WLPzALW+idbB/Z0HpeJNzu2eT0nLlIJ2faX
eTPiMj5QpJSK90Kug1F133cVJJ51Ebm4uo0aN4tOf/jTbt29POMc0TZYtW0ZZWRk+n49FixaxdevWhHMCgQDXX38
9xcXF+P1+Lr74Yvbt29f/2aSJPF+7nlKG0ZYtpaCUiGQArRW9U2aX78V7SgEYBhvHXYVZfCw0VsAfvwbRSMLr6lt
DbN5vZTQ9ePk88n1uXPV7rCc7Kd176NWP2Li3lrwsF+ctPdc6WL7R6kNlm1GWh8OAg/UBDta3dngPEZFk0VrRdyW
5VrPzg/UBao6aKWU3OR/ptw4kY/e9mPELrK973wLaglIf29lZIiI90aug1MqVK/nWt77FmjVrWLFiBeFwmKVLl9L
U1PaL55577uG+++7joYceuYceZSWlrJkyRIaGtqa791www08++yzPP3006xatYrGxkYuvPBCIpFIZx875LSV74W
sA2p2LiIZRGtF74y2y/caAmErw9YWcXoJX/IYuLNhx2vw+n8mvO6dXdVETZhYlM2xJbl8/YzJjDOsjKpI/viEczf
sreXBVz4C4EefnsnISXPB4YbWWqjdEz8v2+OK78a3cW9tcicqItKO1oq+i2XYVja0y5Sq3QNNVR3OjQWIpozMsZq
hB+0/u9yS/g8klim17x0IB61G6ihTSkR6p1dBqZdeeomrrrqKGTNmMGfOGfOHB577dDH27NnDOruHhWmaPPDAA3zve
/kksuYebMmTzxxBM0Nzfz1FNPAvBXV8ejjz7Kvffey6233sqPP/7Ivffey6233sqPP/7Ivffey6233sqPP/7Ivffey
EIiKDR2tF7/i9LvLsdaO87ojspJHT4IL7rMev3QU7VsafWrOjGoBTJhcBcNXCiUx2HQZgfX1e/LzmYJh/eWYDkaj
JRXPK+NTcMeDyQMl064Qu+kppBz4RGUhaK/pulN1T6mB9K2GXH3PEFOuJA+92ODehfC+28ZInB7y5/R9I8bHgGwH
hFqjYxDF2plRFfSuNgXD/319EMoKrPy+uq7P+wTpixAgAdu7cSUVFBUuXLo2f4/V6OfPMM1m0aBGKonDeeedx
KJZxTVlbGzJkzWb16Neeee26zHzwkEAgQCbVuL1tdbOwuFQiFCoVCH81PN77ZifXXNQUKhEI7skTiBSF15fLzpOO5
kyYQ5guY5nGTCHCF189NacXSj87Oob21kb1Ujk0ZkJf5MzrgU585/4Nj4a8w/fI3wP70O/mJWf2QFoE6cUEAoFML
jgBm+GmiBX71vcHxLAK/LwR0vbGPn4SZK8rz84IJp8fd2lszCUb6RyP53iU49Lz6WGaOtC5UNe2sG9M8t0/6/0zy
HvkyYI2itGApGZluXcAfrW6EQIqVzcFV/TGTvWqITz4yf1xQIx292jC/wEj60Dxdg5pQQTtJcnWNPxvHhS0R2riK

7ZA7FOR4ONwbZfqA2fpOjPzLt/zvNc+jLhDlCcufX56CUaZrceOONnH766cycOROAigorE6ikJDEdtKSkhN27d8f
P8Xg8FBYWdjgn9voj3XXXXdx+++0dji9fvpzs7I67C6XangoDcPLhrr28+OJuji2v43hg73vr2Ni0AoAVK1akdIy
DIRPmCJrncDLc59jc3Dzon6m1omecAQfgYPmqtTR+2NbjKfYz6WQRZ2a9Sm7TAd77/Y/ZWriErQecgEHTjnd5cf+
7YEa5sHU/AO825rPsV3+jyAtPve8E4LNjmnnj1baf8YmHncwBDm96mTXNc+PHGxoBXKzfdZi//OVFDGNg5z7c/7+
L0TyHj+E+R60V6W+P/Xt6z6F6mADv1fmYBVRu+BtvN0yPn7fXPi/HZbL6tRWMqX6TE4HDQQ+rX3wxKWM5pimfGUD
lO8/xdvUkChxODmPw7N9Xs2+kedTX99Rw//8uRvMcPob7HJO5VvQ5KPXtb3+bTZs2sWrVqg7PGUf8C9Y0zQ7HjtT
dObfddhs33nhj/Pv6+nrGjRvH0qVLycvL6/Q1qRTacIDf79yCv7CY888/EWP9IfjrHxk/wsuoJUtYsWIFS5Yswe1
2p3qoAyIUCg37OYLmOZxkwhwBqqo69poYaForeubN8Da2rd3HyPFTOf+cYzr9mXQU7ICVdzEzp47yaSdhrn2XiUX
ZXP6Z0603qS/HuSFM1HBSbhax8pDPfvcgV546nn85/7iEzzT2l8DjTzAqcoDzzzuPWPQpGI7yX9v+TnMYZp66iAk
jBuYiLVP+v9M8h49MmCNorRgKKhsC3Lt5JQ1hg6gJU8+8DJ76NaWR/Qm/z5/bWA6bN3PcmELOP/9kHGt2wm4omjC
d888/PyljMfaNhCeeoTS0i/PPO4814ff4aO0+csqO4fwlU/v9/pny/53mOXxkwhwhuWtFn4JS119/Pc899xyvv/4
6Y8eOjR8vLbW2Fq2oqGD06LYdHSorK+N3OUpLSwkGg9TU1CTc1aisrGThwoWdfp7X68Xr9XY47na70/IvujDHqvO
ubw1b48svA8DRVBkfb7qOPZkyYY6geQ4nw32Ogz03rRU9N7bQCvwcbAgmjDVh7JPPgJV34dj7Jmv9NQCcOqWo7fl
GK0vKyBtDqTeH/bUtABwzKofbzp+O2+1M/NAxc8BwYjQfxt16GPLK7M+E6WX5bNxby7aKJo4p6X/5RXeGwt9PMmi
ew8dwn6PWivRXWuDC6TCIRE0aQuAcO8/6fd5UibvlEOSPAWB3jVW6d8yoXGtuzdZmGI78MhzJmuu4+eDKwmiuwl3
zEVNLrMDezqrmpP55DqW/n/7QPIeP4T7HZM6tV43OTdPk29/+Nn/84x955ZVXmDRpUsLzkyZNorS0NCFVLRgMsnL
lyvjCMH/+fNxud8I55eXlbNmypcvFY6iJ7aR0ILa9d7zReWWKRiQiMni0VvReqb1udGh03t4Y6x/+NB1i30dWc/J
Yk3MAaq1yFqNwAv+82Lo77XIYPPD5uWQdGZACcPtgpJ09dUSz8zmxZufagU9EBojWir5zOgxG5liBtbog1i6to46
3njywPn5eQpNzgIZy62tuW5Cv31xemHSG9XjLHzhGO/CJSC/1KlPqW9/6Fk899RR//vOfyc3Njddq5+fn4/P5MAy
DG264gTvvvJOpU6cydepU7rzzTrKzs7n88svnNFBUVMWLECG6++WZmzZrF4sWLkz/DFBg3wrq4qG4
K0tAaIje25WrjQTCTV1stIpKOtFb0Xpm9vfeBupauT3J5YexJsOsfFB1eCyxODErVWEEpCify2RPGcrgxwDEjc5g
5pptMp9FzoHKrFZSa1tbsfPbYAmC3duATkQGjtaJ/SvK8VNS3Uhe0yxTL5sHBLdYOfMdfBMCOQ00ATB7pt85psPt
s5ZYmdzBzvgAfLodNzzDlBKs0cndVM6FIFLezVzkQIpKBehWU+vnPfw7AokWLEo4/9thjXHXVVQDceuuttLS0cN1
111FTU8OCBQtYvnw5ublt247ef//9uFwuLrvsMlpaWjjnnHN4/PHHcTo7uZM7BOVmuRnh91DdFGRvdQvTR420noi
GoLU2pWMTERloWit6b3SBnSlV24rZ3c2LCafBrn+wwPEebxZ+ihJ7W3AAanZZXwsn4HQYXLfomB588BzY+FSXmVJ
bDtQRiZo4HQPc7VxEMo7Wiv6xfv/XWZlSAGNOgHf/D/ZbmVLRqMmOrjKlcpIclJp2PnjzoG4vo2vWk+1x0hyMsLu
qOZ45JSLSlV4Fpbr9h7LNMAyWLVvGsmXLujwnKyuLBx98kAcffLBkffLA3Hz+kjBuRTXVTkD3VzUwvywNfIbTUqIRPRIY
9rRW9N9rOlGoJRahrCeF3dxEEmngarIQFjvc4ZfKIxOfs8j0KJvbig+dYX48ISk0emYPf46QpGOGjykamleZ28mI
Rkb7TWtE/sZsSNe0zpcDKlDJN9te2EAhH8TgdjC30WdUaA5Up5fbB9E/Bu/+HY/PTTB75Bbbsr+fjQ40KSonIUSm
fcoCMK7Tueu+rsbdKtPtKGY0HUzUkERFJU1luJyP8HqBdP8LOjD2JEC5KjFrOKTmiX0e8fG9Czz+4dCZgQP1+aDw
UP+x0GPGyv437anv+fiIiMihiNwu219pBqVEzwOmxqjKqd7DjsFW6N6EoG5fTAYF6CNnXJckOSoFVwgew9c8cX2Q
1QFZfKRHpCQWlBsh4ewvtPdWJQSmaFJQSEZGOYtlS5d30laoNOdkQnQLASWxreyIctAJLAAW9CEp5c6HILvM7soR
vXAEAmxSUEhFJO5+cWYrTYbC3yWBXVRO4PFA6y3rywLt8XHlk6Z59DeLNB48/+QMavxDyx0OwgXOMd4C2RusiIt1
RUGqAdBWUMlS+JyIinRjdgx343t5ZzZqotcNS3sG3256o2wuY4PJBzqhefnCshG9DwuHZsR341OxcRCTtFOd4WWi
XcT+/yS7LKzvB+rp/fTwg1NbkPLbz3gBkSQE4HDDn8wCcULccIB4YExHpjoJSA2ScHZTaGw9K2RcJTQpKiYhIR2U
FR8+UWrOjmrfsoBS73mjb0bVdk3OMXjYl76Kv1JyxBQC8V15PIBzp3XuKiMiAu3C2FWB6YVOF1aNrjB2UOvBufOe
9tkypAeon1d5sq4Rv5MFVFFPHx4eaetQ7TEQym4JSAySWKbW3poVo1FRPKRER6VZprHyvm55Sb+6oYl10KlHDBfX
72pqbx5uc96J0L6ZsrvX1iKDU2EIfhdluQhGT98obev++IiIyoJYcX4LLMNlxuIlt5fVtmVLlG9lZaWW5Thl1xM5
7uaMHbkDFx8CYEzHMCJ9yraYxEKayITBwnyciw4KCUgNkdH4WTodBMBy1fhnH7koMcvneGx8d5urH3qaim3IQERF
JvTK7fO9AF5lStc1B3q+op4UsIqVzrYO73rC+9qXJeUzpbPsDdlu7xNoMw4j3ldqwp6aTF4qISCrlZrmYUWhlIj2
34QAUTwW3H0JNjG3aArQv34tlSpUM7KDshuef91jrk5qdi8jRKCg1QFxOB2MKrAuMPdXN8fI9YxDL90zT5D/+vIV
Xtx/i12/tHrTPFRGR3mtrdN75TYS3dlZjmnDMqBzcUz5hHdxtB6X6kynlK4DCidbj8k0JT82NBaX21vb+fUVEZMD
NL7aCUs9vPEAUB0xYCMBvPD/mzuynyIvama6DkSkFMOMScLg5NrqDY429anYuIkeloNQAGjfCCkrtrW5u231vEDO
l1u+p4WO7nlwXFCIi6a2soK3ReWc9ONbsqALglMkjYMLp1sFdq6yv/cmUAiibZ33d/teEwwpKiYikt+mFJjleFwf
qWnlndw1ceD8Voz6B24hwefQF+Ok8WPNze0MMBranFIC/CKYuBeAS5yplSonIUSkoNYASduCL9ZRqqcaIhgfmA5s
OtzW9BX67dl/88Ya9tVZvKxERSUsleVkYBgTDUaqbggnPhSJRXtpilV4snFIM4xeA4bQypOra9ZaKZTz11rwrrK/
rfwXN1fHDsaDUrqpmao4Yk4iIpJ7bAUunWxUZz23cDwXjeHLKvXwl+F0qsiZDay289K+wf531goHOlIJ4Cd+nnav
YUakdXEWkewpKDaCEHfh8heBwA+ANJ/mXc0stPPsN+H9T4JU7AGgKhHlh04H4KQ2tYXYc1p0KEZF05XE5KM7xAlB
el9gY9m9bKyiva6XI7+Hs40aBN7dt17wPXoJmK4uqT+V7AFPOgdJZEGqCtf8TP1yQ7WFSsdWPZMO+2r69t4iIDKj
YLnwvbq4gFIny8aFGXo/O4cWFv4WL/gv8I9tOzisb+AEdey5hTz6lRg0FB9cM/OeJyJCmoNQAGlcY24Gv2dqi286
WygolMSj18Svw84Ww8TfW99v+BMBfNpfTFIwwqdjPSRMLAVi/pzZ5nysiIklXZveVqqhP7Cv12Bu7APjSgvFkuZ3
WwYmnWV83PGV99RVCVl7fPtgw4LQbrMdv/QKCzfGn4iV8WkNERNLSqZNGUJzjobopyKqPDsf7OE0uyYf5V8H16+G
sf4ezvgcF4wd+QC4vkeM/A8BZgVdoDUUG/jNFZMhSUGoAJZTvQbzZeVIypYJN8Jeb4P8+A/X7oXASGA6o+gjq9vP
btVbd+OdOHMsJ462glHqCiIikt9H5bX2lYjbtq2Xd7hrcToMrTmmXCRXrKxUryehrllTM9E9b79FcBe8+GT+svlI

```
iIunN5XRwwSyrLO9P7+5n12Hr2mPKyBzrhKw8OPMWOPPWQRuTZ/4XATjX8Q7lNU2D9rkiMvQoKDWAYkGpg/UB6w5
BPFOqtn9vvOct+MXpbSUWJ/0TfPMNGD3X+rzNK3hndw0OAz57wljmjS8A4F3d5RYRSWulnezAF8uSumDWaEblZbW
dPP4UwGj7vq9NzmOcLlh4vfX4zQchYvU/jAWlNu6r7bQBu4iIpN7Fc62yvBc2lROMRPG6HPENNFLBGHsSrXjIMVq
p2ft+ysYhIulPQakBVJDtJtfrAmBfTXNbplR/yvfq9sOvLobqHZA3Br78LFzwn+DxwyRri/CKjSsAOGvaKEryspsh
nZ0ptr6inKTBATdZFRKTfygoSg1KVDYF4f8CrT5uUeLKvwOoDFdPXJuftzbsCsouhdg9sfRaA40fn4XE5qG0Osau
q+ShvICIiqXDC+ELGFvqI2BsbTSr243QYR3nVAHI42e+x1q3A/g2pG4eIpD0FpQaQYRiMjTc7b4ER1i/m0XXrwIz
27U13r4ZwKxRPg2+uhilntz036QwARh1aA5h87sRxgLWj0+j8LKImbN6vHTBERNJVrHwv1lPq6bV7CUVMThhfwBw
7YynBxNPbHve3fA/A7YNTvmE9fuMBME08LgczyqxeVRv21vT/M0REJOkMw+CiOW1NzOOleyl0OGcaAM7KrSkeiYi
kMwWlBtj4EdYFxp7qZpj3FUxPDgUtuzHshuS9VrHJ+jrpE9Zd8oQPO5Wo4WI0h5njr+Wc40fFn1IJn4hI+mufKRW
OwlNv7wM6yZKKmXBa2+P+lu/FnPQ18OTAwS3w0cuAmp2LiAwFn5rbPijlT+FILM2F0wHIq30vxSMRkXSmoNQAS2h
27i8iesq3AXCuvAvCwd6/YcVm62v7ko0Yj5+PvccB8PVx+3A72/56542zSvje3aO73CIi6SqWKXWwPsC6wwZVTUF
K87L45MzSzl8wYWHb48IuAle95Su0dmsCWPUAoGbnIiJDwXGleUwryQVgWmkfd2NNptKZAJQ0f5jigYhIOlNQaoC
Ni5fvWX04ogu+QasrH6NmJ6x/ondvZprdBqUqG1r5a9OxAJzh2pbw3NxYptReNaoVEUlXo3K9OAwIR01e2mct0V8
+dULCTYYE2SPgkz+BM26BEZOTN5BTrgOHG3avgr1r4zc2tpXXEwhra28RkXT10y/O43vnH9/1zYxB5Bs3i6hpUBi
thsZDqR6OiKQpBaUG2Lj2mVIAnhw+KP2U9XjlPRBo7PmbNVRA82EwHDBqeoenn12/nzfCMwDILX/TCmLZZpbl43I
YHGoIcKDdrk4iIpI+XE4HJfYOe9UBA6/LwRdPHt/9i075Jpz972AksaFt/hiY/Xnr8RsPMG6EjxF+D6GIybYD9cn
7HBERSapppbn80xmTU9vk3DZ65Eh2mdbu42bsxrqIyBEUlBpg49tlSsUylHYVLcIsmAhNlbDm5z1/s9gv8+JjrWa
07ZimyW/f2cu75jGEHV7rvQ9tjz/v8zg5frSVxqsSPhGR9DU6Pyv++OI5oxnh96RmIKd9x/r6/gsYO15TCZ+IiPR
KaX4W75nWjZXmPe+meDQikq4UlBpgYwp8GAY0BSPUNIcAMB0uIotus05447+gqapnbxZrct5J6d47u2v4+FATTnc
WjD/FOrjz9YRz1KhWRCT9jS5ou+lw5SlHyZIaSCOnwbwrrMe/vZKziqwbGgpKiYhIT3hdTna7pwAQ3K9MKRHpnIJ
SAyzL7aQk17rrHS/hA8zpn4HS2RBsgH/c27M3iwelZiccjkRN7njB6iF10ZzRuKacaT2xc2XCefPa9ZUSEZH0NNY
OSk3NizKtNDe1gzn/Xhh3CgTquPT9GymiTkEpERHpscM50wBwHdqS4pGISLpSUGoQjD+yrxRYfaEW/8B6vPaXULu
nw+sC4QhNgXDbgS6anD++ehcb99WRm+XipqXTYJIdlNq1CqJtDWnnjbca1W7eX0cwHO3nrEREZCBcccoELplXxuc
mp8HvaXcWfOEpKJyIr2kv/+25j4qqWqqb+rB7rIiIZJzWEccD4G/YASH1tRWRjhSUGgRH7sAXN+UcmPgJiATh1bs
SngpFolz401WcfvcrvFdeD4EGqN5hPdkuKLW3upl7l1u9o24773irQe7oueDJhdbatkAWMLEom4JsN8FwlPcr1Kh
WRCQdjRuRzd2XzKTEd/RzB4W/CC7/HWTlM9/xIf/P/Qgb91SnelQiIjIEZBeNo9rMwWFG4NB7qR6OiKQhBaUGwbg
R1pVFh6CUYcDi263HG3+T0Jh8+daDfFjZSE1ziCv/920OfrjOeiK3DPzFgNXc/N//tIXmYISTJ47gCyeNs85xumD
iadbjdn2lDMOI95V6V32lRESkp0YeC59/kghOLna+ie+Ne1I9IhERGQLKCrN5LzrB+kY78IlIJxSUGgSdlu/FjJ0
PU88FTNjyx/jhx1fvBMDjdFDZEODp5/5iPdEuS+q5jQdY+cEhPE4Hd14yC0f7rV8nnWF93fWPhI+bN84q4dMOfCI
i0iuTzuCtmf8BwCn7HoUNv0nxgEREJN2VFfjYZsaCUuorJSIdKSg1CLoNSgFM/5T19YOXANiyv461u2pwOQx+941
TGVPgY1TzhwCERs4AoLopyO3PW83Nrz/7GI4ZlZP4nrGg1O7VEAnFD8+1m52rUa2IiPRW7ilX87PwxQCYf701oW+
hiIjIkcYU+NimTCkR6YaCUoMg1lOqvK6VUKSTxrVTlwAGlG+A+nIeX70LgPNnjWbOuAKe+OrJzHZZjdB/8UEOoUi
UO/6yjeqmINNKcrn2zCkd33PUDPCNgGAjHHg3fnju2AIAdlU1q1GtiIj0ynGjc3mIL9BkejEC9VD1UaqHJCIiaay
sIIv37Ewp8+BmMM0Uj0hE0o2CUoNgZI4Xr8tBJGpSXtfJrhM5o2DMfAAatrzIcxsOAHDVaRMBOKbIy/HOvQD8bn8
hX/rlW/xx/X4MA+767Cw8rk7+Gh0OmPQJ6/HOlfHD+dlupoz0A7Bhr0r4RESk59xOB9PHFPK+Od46oLveIiLSjRF
+D/ucYwmYLoxAA9TuTvWQRCTNKCg1CBwOo20HvpqWzk869pMAHFr3J4KRKHPG5jPPbkrO4Q9xRIKEXX4OGKN4e5e
169FXTpnACeMLu/7gWAlfu2bnAHPtvlIb1OxcRER6ae64gnalGJtSOxgREUlrhmEwqiCXD82x1gHdzBCRIygoNUh
ifaX2VncRlJpmBaVGV72FlyBXnzYJw7Abl9u/vF1ls7nrkjmAVZ99yyeP6/5DJ9pBqT1vQagtQ2ue3VfqXfWVEhG
RXpo7roCt5kTrm3IFpUREpHtlBT7ei8YybNXsXEQSuVI9gEwxrtAHwL6aFqZ3dkLJTFp8pfhaKvik/0POn/Wptud
id6JLZ/G5E8dxwoRCinO85HiP8tdXPBVySqGxAva+BZPPBNqCUhv21BKNmom79omIiHRj7rgCfmlnSpkVmzFMEwy
tIyIi0rmygqx2O/ApU0pEEilTapC0le91sQOfYfA6JwDw1VHbE/tExX55l84CYMrIHPJ97qN/qGHAlLOsx1t+Hz8
8rSQXn9tJQyDMx4caezcRERHJaGMKfOx2TSBsOjCaD0NDRaqHJCIiaayswMe26ETrGwWlROQICkoNkvFH6Sm1aV8
tT9fNAGBm45q2nSlMs11QanbvP/iEK+0P+B00W72oXE4Hc+1+VWt3qdm5iIj0nMNhMLqokI/NMuuALjBERKQbZQU
+3jPHWd/U7YGW2pSOR0TSi4JSgySWKbWvi6DU42/sYnV0BkHDi7NhH1Rus56oPwAt1eBwwcij9JDqzPhTrGBWuAX
W/yp++KSJVrPzd+ym6SIiIj01eaS/XSnGxtQORkRE0tqYAh/15FBhjLIOHFRfKRFpo6DUIIkFpWqaQ7SEE5+rbGj
l+U0HCOChZezp1sEPXrK+xvpJFU8Dd1bvP9gwYMG11uO1j0I0AsBJk0YAxHfyExER6alJxf52O/ApU0pERLpWVmD
11t0ab3audUNE2igoNUhyvC6K/B4AqgJtx3cebuJHL7xHKGIyb3wB+XMutJ744G/W1yP6SfXJzM+Cb4SVLrv9rwD
MG1+Iw7Ayt8rrutgRUEREpBOTinPUtFZERHpkdL51Y31LRDvwiUhHCkoNoniz80aDp97eyyUPv8FZ//kaz288AMA
/fWIyTD3XOnnv29BUlbDzXp+5fTD/KuvxW78ArCDZjLJ8QH2lRESkdxIypap3QGt9agckIiJpK8vtpMjvaZdhuym
1AxKRtKKg1CCKBaWe3uHkB8+/x/o9tTgMWDRtJI98eT7nzxoN+WPsAJQJHy5PTqYUwEnXgOGEXf+Ag1a/qhPtvlJ
rd6qET0REem7KSD815HHAtErBObg1tQMSEZG0VlbgY5tpZ0odeh8iodQOSETShoJSg+i40tz442klOXzv/ONZc9s
5PH71yZw7o7TtxGM/aX3d/Fuo2WU97m9QKn8sHG+XBr79CAAnT7QuJtaqr5SIiPRCQbaHwmy3+kqJiEiPlBVksc8
cSdCVA5EgHP4g1UMSkTShoNQguvq0ifzkMzO4dXaYF769kH86YzKj8jppXn7sedbXj1+xvuaPg+wR/R/AyXbD802
/hZYaTrSDUtsPNlDXorsVIiLSc5OK/Ww1J1rfaAc+ERHpRlmBDxMHFb5jrAO6mSEiNgWlBlG2x8VnTxjDGP9RTiy
```

bB/6Rbd/3N0sqZsJCKJkJoWZ490lG5nqZVOzHNGH9bvWVEhGRnptUnKNMKRER6ZEx9g58O52TrANaN0TEpqBUOnI
42hqeQ/KCUoYBC+xsqbf/G6IRTpxg9ZV6WyV8IiLSC5NH+tt24Kt8T/1BRESkS2V2UKptBz41OxcRi4JS6erYAQh
KAcz6HPgKoXYPfPA3TppklfC9o6CUiIj0wuRiP3vNUTQZ2eoPIiIi3YoFpd5uGWsdqNgMppnCEYlIulBQKl1NOQt
cWYABo+cm733dPjjhK9bjtx/hJLuv1Ma9dbSGIsn7HBERGdYmjfQDBu/FsqXKdddbREQ6V1Zg9dF9q2kUpsMFLTV
Qty/FoxKRdKCgVLry5sLlv4XLfgUF45L73id9DQwH7HiNibVrKM7xEoxE2by/LrmfIyIiw9bEIqtB4uZwrBRD/UF
ERKRzxX4vHqeDVtNNaMSx1kGV8IkICkqlt8lnwvSLk/++BePhhCsBMH5/NeeXNQHw9k6V8ImISM9kuZ2MKfC124F
PFxciItI5h8NgtJ0tVZd/vHVQGbYigoJSmeu8u2HsSdBax79ULSOHZvWVEhGRXplU7G+3A98m9QcREZEuleVbfaX
KfcqUEpE2CkplKpcXPv8k5JZR2LSDB9w/Y/3uKiJRXVCIiEjPTB7p50NzLBHDBa11ULc31UMSEZE0FWt2/pFzsnV
AmVIigoJSmS23FL7wJKbTy2Lnu3w9/BQfHGxI9ahERGSImFTsJ4SLA+5YtpT6SomISOfG2OV7WyJ2L8L6fdCsSg2
RTKegVKYbMx/jUw8B8C3Xcxx689cpHpCIiAwVk4qtZufb4n2lFJQSEZHOxTKldjY4oHCSdbB8YwpHJCLpQEEpgdm
X8c7YrwBwyuYfwIENqR2PiIgMCZOLcwBY2zrGOqBSDBER6UIsKHWgthVGz7YO6maGSMZTUEoACJ7577wSmYvHDGD
+7iqIRlM9JBERSXNjCn14nA42h1W+JyIi3WsLSrVAaSwopZsZIplOQSkBYN6EYm6MXE+D6cOo2QkH3k31kEREJM0
5HQYTirLZZtpBqbo90FKT2kGJiEhaKrN7SjUEwjQXzbAOKsNWJOMpKCUA+DxOJo4ZzcroHOvA9r+kdkAiIjIkTCr
200A2DT67hE/ZUiIi0olsj4vCbDcA+31TrYNVH0KwOYWjEpFUU1BK4k6eNILlkfnWN++/mNrBiIjIkDBppNXsfK9
ninVAQSkREelCrIRvbzAX/KPAjMLBrSkelYikkoJSEnfihEJei84ljBMOvQfVO1I9JBERSXOTj9yBT6UYIiLShVh
Qan9Cs3PtwCeSyRSUkriTJo6gHj9rIsdZB5QtJSIiRzHJ3oHvzVa7r9Suf4BppnBEIiKSrsZ01uxcNzNEMpqCUhJ
X6PdwbEkOK6InWge2KyglIiLdm2yX773QMAXT7Yf6/VC+IbWDEhGRtBRrdn6gtqVdppSCUiKZTEEpSXDypBG8HDn
B+mbPm9BUldoBiYhIWivye8jNchEwPTSOPcM6qExbERHpxPgR2QBsO1Dflil1cBtEQikclYikkoJSkmDBpCL2M5K
PnZOtxoMfvJTqIYmISBozDCPeV2pn8ZnWQWXaiohIJ06dUozLYfBhZSM7o6PAkwuRABz+INVDE5EUUVBKEpw8aQQ
AzwfmWgd0YSEiIkcxyQ5KrfWcBIYTDm6Bml2pHZSIiKSdfJ+bU6cUAbDivUoonWU9ob5SIhlLQSlJUJKXxcSibFZ
E7L5SH78CoZbUDkpERNJarNn5+7VuGH+qdXD7X1M4IhERSVdLppcAsHzrwXZ9pTancEQikkoKSkkHJ08awVZzAnW
eUgg1w47XUj0kERFJY7Fm5zsPN8Fx51sH3/9LCkckIiLpavHxVlBq3Z4a6gumWwfV7FwkYykoJR0smFQEGKxynmQ
d0IWFiIh0I1a+t+NwE0yzg1K7V0NzdQpHJSIi6aiswMfssfmYJqxuKrMOVmwC00ztwEQkJRSUkg5ifaWeqbdrvD9
4CaKRFI5IRETSWSwoVd0UpDZrDIyaDmYEPlyR4pGJiEg6WmqX8P1hbw44PdBaB7W7UzwqEUkFBaWkg7GFPsrys1g
dOY6wOxeaDsG+d1I9LBERSVN+r4uSPC9gl/DFsqW2K9NWREQ6WjqjFICVH9cSGXm8dVDNzkUykoJS0oFhGJw8aQR
hXHyUH2tYqwsLERHpWixbKqGv1Ed/h1BrCkclIiLpaOqoHCYWZRMMRzmQNdU6qL5SIhlJQSnp1MmTrK1aXwrPtw6
8/2IKRyMiIulu8khrB76dh5tg9DzIHQ3BRtj5eopHJiIi6cYwjHi21FstY62DypQSyUgKSkmnFky2+ko9cXgqpsM
NVR/C4Q9TPCoREUlXk+1MqY8qG8HhgGnnWU8o01ZERDqxxO4r9dxB67pDmVIimanXQanXX3+diy66iLKyMgzD4E9
/+lPC86ZpsmzZMsrKyvD5fCxatIitW7cmnBMIBLj++uspLi7G7/dz8cUXs2/fvn5NRJJrcrGf4hwPNeEs6ktPsQ5
qFz4R6SGtFZlnzrgCAF7dXklVYwCOu8B6YvtfIRpN3cBEJG1prchsJ4wvpMjv4Z3WsZgY0FAOjYdSPSwRGWS9Dko
1NTUxZ84cHnrooU6fv+eee7jvvvt46KGHWLt2LaWlpSxZsoSGhob4OTfccAPPPvssTz/9NKtWraKxsZELL7yQSEQ
7vKWLWF8pgA3ZC62D77+QwhGJyFCitSLznDihkDlj82kNRXnsjV0w8RPgyYXGg3BgfaqHJyJpSGtFZnM6DBYfX0I
zWVR57RK+io2pHZSIDLpeB6XOO+887rjjDi655JIOz5mmyQMPPMD3vvc9LrnkEmbOnMkTTzxMzBc3MzTz31FAB1dXU
8+uij3HvvvSxevJh8x5JNPsnnzZl5++eX+z0iS5uSJVlDqd81zAQP2rYUabdUqIkentSLzGIbBNxdNAeCJN3f
REHbA1MXWk8q0FZFOaK2QpTOsEr53Q+OtA+pDKJJxXMl8s507d1JRUcHSpUvjx7xeL2eeeSarV6/m2muvzZd26dYR
CoYRzysrKmDlzJqtXr+bcc8/t8L6BQIBAIBD/vr6+HoBQKEQoFErmFAZcbLxDYdzzx+cD8Mo+B5GJp+Hcs4rI5t8
TPfU73b5uKM2xPzTP4SMT5gjpMz+tFUc3VH8mz5paxORiPzsON/Gr1Tu59phzcW19FvP9vxA+898Szh2qc+wtzXP
4yIQ5QvrMT2vF0Q2Hn8mTJ+ST7XHyh9aTWOJ5A3PNzwnP+gIUWTvyDYc59oTmOXxkwhwhufNLalCqoqICgJKSkoT
jJSUl7N69O36Ox+OhsLCwwzmx1x/prrvu4vbbb+9wfPny5WRnZjnZydj6INuxYoVqR7CUUVN8DmdNAcjvNY6lXNYRcO
bT7Cy5pgevX4ozDEZNM/hY7jPsbm5OdVDALRW9MZQ/Jk8Jd9gx2Enj7z6AWPnmFyEE8fh7ax89jGavCUdzh+Kc+w
LzXP4GO5z1Fox9Az1n8mpOQ5eqj6JLe7ZzAxtou7Jq3njmNvAMOLnDPU59pTmOXwM9zkmc61IalAqxmj3CwSs9Ns
jjx2pu3Nuu+02brzxxvj39fX1jBs3jqVLl5Xl9f/AQ+iUCjEihUrWLJkCW63O3O9XDOarna97lle2H2HvMlzAP/Yq
Clt2cv+BYKOo6MDXU5thXmufwkQlzBKiqqkr1EBJorejaUP6ZXByO8uoDqyiva6Wu7BRoPA12vc5Zo2qJnnZ1/Ly
hPMfe0DyHj0yYI2itGEqGy89kqOwAG/+whXu83+YJbqC48X0uGFOLOfdLw2aOR6N5Dh+ZMEdI7lqR1KBUaWkpYN2
1GD16dPx4ZWVl/C5HaWkpwWCQmpqahLsalZWVLFy4sNP39Xq9eL3eDsfdbveQ/YseKmM/ZUoRr2w/xKpyuGrKWfD
Ry7i3Pw9n3nrU1w6VOfaX5jl8DPc5psvctFb03FAcu9sNXz9jMrc/v43/WbWbLy3+Iux6Hefbv8B56nXgzTni/KE
3x77QPIeP4T7HdJmb1oqeG8pjB1gyowzns1t5/XAONWffxIjVP8L19x/A8ReAtwAY+nPsKc1z+Bjuc0zm3Hrd6Lw
7kyZNorS0NCFVLRgMsnLlyvjCMH/+fNxud8I55eXlbNmypcvFlFALy9s5roDLsJ5ZY/pHBEIjLUaa0Y/j5
/0jgKs93sqW7mL5wGIyZDcxW8/d+pHpqIDBFaKzJHfrabBfau3791XQSls6C1Fv52W2oH1gMfH2rk20+t542PDqd
6KCJDVq8zpRobG/noo4/i3+/cuZMMNzdJ1KlTmTp1KneEeeSfZ2dlcfvnlvnAOTn53PNNddb2vIKSolIF7R

WyJWnTuSRlR/zfkUDr7nP5KyiqVD1Ibz9CJxxS6qHJyJpQGuFxHxq7hjufmk7+2tbWFF7POct+AaseRjnS7fgnPD
vqR5eB1sP1PHlR9+muikIwKZ9tbSGImS59XMn0lu9Lt9btGgRpml2+O/xxx8HrGaEy5Yto7y8nNbWVlauXMnMmTM
T3iMrK4sHH3yQqqoqmpubef755xk3blxSJiTJF0unfX7jAcLTP2Md3PIHMM0UjkpE0pnWCsnPdvOlUyYA8PDrO2H
Rv1pPrH4QWutSODIRSRdaKyQmy+3kSwvGA/C/b+yEs74HeWMxanczrfxPqR1cTLAZwkE27avl8l++RXVTkNlj8yn
O8RKKmGzYW5vqEYoMSUntKSXD03mzRpPldvDO7hpu21qG6fJBzU448G6qhyYiImnsmtMn4XE6WLurhrezz4SRx1k
BqTU/T/XQREQkzVxxygTcToO1u2rYdCgMF/wnAFMq/4qx7+3UDcw0Yf2v4J7JND98Jl/75UrqWkKcML6AJ7+2gAW
TrRv4a3dWp26MIkOYglJyVFNG5vDzL83H5TD43aZatuacaj2x9Y+pHZiIiKS1krwsPjt/LADLXnif8BnftZ5482F
oqU3dwEREJO2U5GVx4ewyAP531U6Ydh7RGZfgIIrz91dB3b7BH1SgEfOPX4fnrodwC9nV2/hu9L85eWIhv7pmAXl
Zbk6eaAWl3t6loJRIXygoJT1y1nGjuPeyORgGPFg52zq45VmIRlM7MBERSWs3LjmWgmw328rr+UXlDBg1AwJ1ON5
WtpSIiCT66mmTAHhhUzkH61uJnH8fdVnjMJoq4TdfhGDTgI/hsTd2ctGDq7jqJ4+x866TMDb/lohp8GT4HMKmg88
6V/Hk/I/I8VrtmU+yg1Lrd9cQjujaSKS3FJSSHvvU3DH88OIZvBadS4Ppg/p9kMpUWhERSXsjc7384CJrY4yfvrK
D8nn/DIDj7UdwhxtSOTQREUkzs8bmc/LEEYSjJv/35m7w5PDWlH/BzC6Gik3wp+sGtK/ty9sOcvvzWzm+4k/8ouU
WJnGACrOQLwS/zx3GP/HSyK8C4PnbrVD5HgDTSnPJzXLRFIywrbx+wMYmMlwpKCW98uVTJ/LtJTNZHp0PwEev/ir
FIzqKcNBK9T24FXavhu0vwcZnrLrwxkOpHp2ISEb49NwxnDVtJMFIlG+vL8MsnYURbOSYypdSPbQu/W1rBSf/+GU
ee2NnqociIpJRvnr6RAB+/dZuWkMRWjzFRC59HBxu2PYnWHnPgHxuXXOI2//4Dve6f8497l+SZYRoGncmzutW8eS
PbuD9H53Hhdf9P5hyNoRb4LdXQrAJp8PgxAmFALytvlIivaaglPTat88+huC0TwOQv+MF/rR+T2oH1JW6fXD/dLh
/Bvx8ITx2Hvzm8/CsXRf+4s2pHqGISEYwDIMff2YWOV4X6/bW83LJNQBMPrQc6stTPLqO/rKpnG/9ej2VDQEeX70
r1cMREckoS6aXMrbQR01ziD9vtNYIc9wpcOF91gmv3Qnb/pz0z33g2Vf5WfB7fNa5CtNwwjk/wH/1nxhZMhavy2m
d5HDAZ/4bckrh8HZ48RYATp5UBMBa9ZUS6TUFpaTXDMPg85//Cs3OXEYadfz297/h/hUfYA5gKm2vmSb85WZoOgQ
OF2QXw4gpUDYPxp5snbPzdfXEEhEZJGUFPv71vOMA+M76UlpHzcEVDeB65gvWjnzJ1HQYNv3Waqj+yo/hLzfB778
Kv/o0PHNFt9myf96wn+t/s55w1FrTdlc1s/PwwPcwERERi9NhcNXCiQA8vnp3W7XeCV+BBd+0Hj/7DSjflLTPfOf
1F7nug68x27GTsLcA48vPwidutIJQR8oZCZc+CoYDNvwaNvyGkydZmVLv7KpJr2sikSHAleoByNDkcHvJmnMJrH+
C77l+zSV/P5adh5u459LZOFM9OID3noMP/mql+X7jHzDq+LbnwkH4yXhoqYaqD2HktNSNU0Qkg1x+8nie33iAt3Z
W813+hbtd/0xW5Vb4zeVwxR/AndW/DwgH4a1fWKUdwW76Vb33PGDAuAVw3Pkw7QIoPoY/rNvHLb/fSNSEz80fy96
aZtbsqOa17ZVMKp7Uv7GJiEiPXXbSOO5f8QEfHWpie7HBBbEnlt4Bh96HHa/C/30GFi+DuZeDo/MrkEjUZMOby4l
sfY5R449j4onnQdEUMIz4Oc1v/i9zXrkFtxGm0jeFUV//IxRO7H6AE0+HRbfBqz+Gv9zIrGtewetyUNUU5ONDTRw
zKicZfwwiGUGZUtJnjkX/CtlFzHDs5nb3r3hu4wEu/+UaqhoDqR1Yax28eKv1+PQbEgNSAC4PjD3RerznzUEdmoh
IJnM4DO7+7Gyy3A7+vMfDI4W3YHpzYfcq+OPXIBrp2xubptUz8OFTYMX3rYDUyONhxiVw0tcwz7iFN465iRtD3+T
e0KVsMScDJuxdAyv+Ax6az4GfXci///5toiZ88eTx3P3Z2Zx93CgAXtuuHoQiIoMpL8vN504cB8DK8rYAEk4XfO4
xayfX5sPw3LfhkTNhx2sJr6+sb+FPf/wN797xCeavuIyTDzzJxDX/Dg/NJ3rvcfDHr8P6/4O/3Ez23/4FN2FWOk8
l79uvHj0gFfOJm2DyIgg143n1R8wdVwCor5RIbykoJX2XVwaf/R/A4AvOV7g8azXr99Ry6SNvUd6cwnG9fDs0Vlj
lep/oom/U+FOsr3vWDN64RESEicV+blpiZaj+4sAk3j75IUyn18pe+suNvd9VqfJ9ePISq2dg9cfgHwWf+hl8czV
87jEazvkJ39x/Hl/aMp8/Rj7Bb3xf4MLAHZza+iDfD13FGmMOEZyUHfoHv3T9J19dUMqPPz0Th8Ng0TQrKLVmRxW
toT4GzEREpE+uWjgRw4BttQ7e3FHV9oSvEL7+qpU15c2Hg5vhV5+Cpz7PhvVv8dAjP2Pvf57Bpzd9gxOjmwnh5K3
sRbwZmU7AdOForIBNz1gBrbW/BOC+8KXkfPlJsvz5PR+gwwmLb7ce71zJKRNyAfWVEuktle9J/0w5G878Lqz8CXe
4HuVA4bG8VlPM/Q1OJs86xNKZZYM7nj1vwTv/az2+6L+6LgWJB6WUKSUiMti+evokXtx8gHf31vH5FW6uL72FG2t
/jLHucat57Fm3JZwfiZq8V15PfUsIE4iaJqYJRXv/xvRV38EwI+D0wCnXWXeus/IA+OBgA9/4v3XsONyE22mw7OI
ZXH7yeNbvqeVP7+7nhU2l/F/zUk4wPuBXnp9wunMrpzXegRH5DTiymDoqh7L8LA7UtfLmjirOsoNUIiIy8CYW+7l
wVinPb6rgn/7vXR758vz4zQJcXlh4Pcz9Eqy8G3Pt/2B88BJzP3iJuQAOCOJmz8RLGXvhv7KgeCLr99Rw2R/XkV2
5noWOrSz2bccdbuLuwKVMPO1zzJ9Y3PtBls62etc2H+bsnN38F8qUEuktBaWk/868FfauwbHjNR7Nf4ircn/CP/a
0cu2v3+XWc1v4xpmTMdrVbQ+YcBCe/2fAhLlXwKRPdH3u2JOt5oQ1u6ChAnJLB358IiICWE1s//fK+dzy2Mv846C
LByumU+28mh+7/xdW/gRyRlI57QpWfnCI1z88zD8+PERtcyjhPUZSy3LvdzGMCK9E5vI/vmspq53BiZtqOXGig23
l9Xz395toCUUYnZ/Fw186gXnjrUa08ycUMn9CId+/cDqvf3CIFzeP4e9ZY7l4y3cwPn4Ffvtl+PyTGC4vZ04bxW/
e3sPK7YcUlBIRGWR3fXoGO/YeYGsN/NOv3uHBL57AJ2e2+3d79ggazrqDO/edwll7HmSpcx0Bh4/GWV+haPGNHNP
u3/gnjC/kD9efzRNvTuG+5bO5t8HKgJ1c7OfBpX3sMetwwJSzYPPvOL75HRzGieyvbeFAbQtlBb7+TF0kYygoJf3
ncMIl/wOPfAJn1Qc8Nv3XfKXl06w+5OTul97n/Yp6u4fIALdAX/1TOPSedbdi6Y+6PzcrD0pmQMVmq4RvxqcHdmw
iIpIgx+viovFRvv+F03jglR38+t3FFFPHv7j/QOQvt3Drswd5LTo3fn6u18XogiwMDAxMljU/QGGokQ8cU7g2cCO
hGhes28fv1+1L+JzTjynmv74wl6Icb4cxeFwOFk8vYfH0EmAuzC6FX38OPlwOv7sKPvcEi6aN5Ddv7+G17ZXAjIH
8IxERkSN43U6uOTbKy02jeXHLQb711Hru/dwcPj1vDAB7qpr52q/W8sFBD3903cIvzvFz1vyZeLNHdPp+LqeDa06
fxAWzRnPHX7bx7p5a7vv83P5dp0w5Gzb/Ds+uV5lRdg6b99exdlc1n5o7pu/vKZJBFJSS5MgZCZc+Bo9fgGvbH7h
1bC4bFnyHH724nT9vOMDOw03895dPpDS/nzsrdaXqY2u3JYBP3gVdLEQJxp+qoJSISIqVFfi4//Nzufq0idzxQiG
l+6r5outVfup+kNsK72Py9PmceexI5o4rwOW0W2FueAr+9DY43Bx77ZO8kzeV9XtqeGdXNWt31bBhby3hSPT/t3f
f4VGU2wPHv7/Mlm957Dt4Tei0DoIKKkwIDCwgKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKIKI
gGogUTbLJcj7PM092Z2dnz7vvPnuyZ2belzuHtubBK9qh113g2bpxg+DGuTD3BjiyBOpPZMDYDzHqFdIKzKTlVxA

b6NF4b4YQQoiz6HXw2vVdcTcd4vudmTzw7R4qa2zEBXpw55c7KTLXEOxl4sNpvel2arDx8wn1cWX2lJ4NE2CrYdr
fE3sY0k1HYpZ2CZ8UpYS4MDLQuWg4MQkw4kkAumR9yU3RhXwxqw9+7kb2ZZYwdvYGdqUXNfzrZu6E76aDrVo7UtH
l+gt7nowrJYQQzUbXSF++uT2B2GnvUBDYG2+lkreVl3hoYBC9Y/1/K0iVZMHSR7Xbwx6DkI74uBkZ1i6Yh69sz7e
3J5D41Ej2PXUlD1/Z/sILUqe1HgY3fKWNUXXoJzw3/IfeMdqBDu1sKSGEEE1Nr1N46bquTEuIQVXhsR8SufmjrRS
Za+gS4cOiewZecEGqwXmHQXBHQGWE62FABjsX4mJIUUo0rIR7sbe5Cr1qxTD/FvqHKiy6ZyDtQrw4WVbNjR9sYd3
RC5haO3MnfDAUFv0d0jaA3X72NvlJ8M1U+Gi4dsaTyQfGvAbnGb8qt7SKTzakku3dTVuRsw+qyy6+rUIIIRqUoig
ktA0n4JZvwTcGilLh22lgOzWelKpqsyVVl0BEL+h/X737MRn0eJr+wsng8SPgmne127u+YHgbXwDWXEj+EkII0Sh
0OoWnx3Xi9iGtALDaVcZ0DePb2xMa72qMC9V6OADtK7YDcDS3nKIKiyMjEqLFkKKUaFg6HbZxb1NuCkEpyYD5M4n
yNTH/rv4Mbx9MtdXOrZ/vYPXhcxxttlpg4R1wYjfs+gw+HQNvdIYVT2jFp7Ic+Ol+eLsvHFoEio6CNhNZNvh7ytw
j/3C3VTU23l6dzLBX1vDM4oM8vKIAfKNBtUPmjoZ/L4QQQvw5HgEw5Rtw8YS09bDkYa0gtfNTOLYKDK5wzXugb8R
RCDpdC54hUFXMKPdDAGw+VkBVja3xXlMIIcQ5KYrCo1e1543J3Xnpuq7MvrEHbi6NPG7thThVlHJNX0vrQHdAzpY
S4kJJUUo0PFcftsX9HdXoDilrYNVzeJoMvHdzL67sFILFauf2L3byy8Hc+p+/eTbkHwWPIOhxs3YGVGmWNpD5ewP
htY6wcw6oNmg7iuTrlpNwaCJ3/HSSPv/5lX98t5ftaYWoqgqAqqosScxmxGtreXn5EcwW7QfF5pQCqsP7aq+ZvqU
J3hghhBAXLLgDXPcxoGjf+b8+DSv+rT02/AkIatu4r6/TQ6cJAERkLCbMx5Vqq50tKQWN+7pCCCHOSVEUrukRwaT
Loppmhu8LEdMf9CYozWJMuHYFhhSlhLgwUpQSjaLMLQrbmDe0Oxteg0OLcTHomD2lJ6O7hGKx2bnzq50s259T94n
F6b8NWD7yORj/NvzjKEz6AjqM077sVRtE9oFbllI24QtuXVqBxWrHw0VPZY2N73dmcv17m7n8tbXMXpXE5A+2cNd
Xu8gsqiTU25U3JnenfagXNrvKfn1H7bVkXCkhhGh+2l0FVzyj3d7wOljKtUkq+t3ZNK9/aoxC5cgSrmjjCcCaI3I
JnxBCiDMY3bTCFHC5ywEAtqU1wli6QjghKUqJRqN2mgD97tLuLLgD8pMw6nW8eUMPxnYLp8amcs/Xu1iSmP3bk5Y
+CtZKiBkAXSdr64yu0HEcTP5CK1DdswNmrUCNTuDRHxJJKzAT7uPKhkeGM//OBCb1jsTdRU/KyQpeWXGUbamFmAw
6/n55G1b9YwjX9Ijgyk6hACwqitZeI3PHb2OWCCGEaD763wvdpmi3je5wzTvaWUxNIaIn+MVBjZnr3BMBWCvjSgk
hhKjPqUv42pZr40odyCrBbLE6MiIhWgQpSonGdcUzWoHJUgbzboLqMgx6Ha9P6sa1PSKw2lXunbubD9elYN6/GI7
8DDoDjHm1/gHL3XwhsA0oCl9uTefnfdkYdAqzb+qJn4cLvWL8eWliN7Y9PoIXJnRhcNsgJvWOZNU/hvLgFW1xd9H
GHzldlPomzQ3V1RdqKrTxqoQQQjQvigJj39DOnp26EPxbNe1rd5kIQKfCFRh0Cqn5FRwvqGi6GIQQQrQMp8eVytp
MjLceq11ld3qxY2MSogWQopRoXHojTJwDXmGQfwTm3wbmQgx6Ha9c342JvSKx2VVeXbKHwu/uByC740zUoPbn3O3
+rBKe/ekgAI+Oak/PaL86j3uaDNzQJ5rPZ/bhpYndiPB1q/N4hzAvIv3cqLLCSb/u2koZV0oIIZong0k7Yyq6b9O
/9qlL+AwpqxgSrR3YkEv4hBBCnCWkE3gEo9SYmRR6AoB52zMcHJQQzZ8UpUTj8wqBSZ+DzghHl8JbPWHrB+hVKy9
d15Vnx3fiCZ+lRConOaH6c/mOfox4bS0frkshvcB81u5Kq2q466tdWGx2rugYwqyBcRcdkqIotWdLbbW201bKuFJ
CCCHOFNQOQrqAvYZpPnsBWHPkHDPICiGEuDQpSu3ZUpP8jqIo8NPeE2xLlQHPhTgXKUqJphHVB6YthOCOUFkESx+
GdwegS/mVqW2sTLEuBOCX6PtRjR4cO1nBf5YcYvDLqxn00ioe+2Efi/edoKC8mke+30d6oZlIPzdemdjtT8+6cbo
o9d3JCG1F+hZtynEhhBDi905dwte77FdAm721qsbmyIiEEEI0R6eKUkG5G7nhsigAnlp0AJtdfmMI8UekKCWaTux
AuH29Nl6Um792Od+X18HHI1BsFogfwbSZf2fb45fz32u70CfWH4NOIaOwkrnbMrjn6930eu4Xlu7PwahXeHtKT3z
cjX86nF4xfgR4uLClKga7zgUq8qAwpQEbLIQQwil0vg4A9xNb6OJVQVWNnc0pBQ4OSgghRLPTepj2N2cfDw/wx8v
VwMHsUr6Ry/iE+ENSlBJNS2+Ay26Fv++Gfndrg5pXFoHeBKNeAkXBy9XIlL7RfHtHAnueHMknM3oza2Ac7UO9anf
z+OgOdIvy/Wuh6BSu6BiCBSPpbqfGsJJxpYQQQpzJNwqiE1BQuTNIu4Tv5WVHsFjtDg5MCCFEs+IZDKFdAPDP3cw
DI9oC8MqKI5SYZaZvIeojRSnhGG6+cNV/4a6tWpHqug8hoPVZm3maDAxvH8ITV3dk2f2D2fb45fz894HMGHDx40j
V5/QlfGsqT722jCslhBCiPqfOlrrCth4/dyMHs0t589ckBwclhBCi2Tl1CR/HVjE1IYY2wZ4UVlh449ejjo1LiGZ
KilLCsQLjtcv5Oo6/oM2DvVzpFO7TYC/fPz4AT5OBtVXx2go5U0oIIUR9Ol0Lih5j7l7eGOEJwDtrktmVXuTgwIQ
QQjQrvytKGXUK/ze2IwCfbz7O0dwyBwYmRPMkRSlxSTMZ9AxtF8ROexttRUESVOQ7NighhBDNj0dg7Q+NIZZ1XNs
jArsKD327F7PF6uDghBBCNBtR/cDgBuU5kHeIQW2CGNkxBJtd5ZmfDqLKxEpC1CFFKXHJu7JTKKV4kqKL0VbIJXx
CCCHqc2oWPhK/56mxHQn1diU1v4IXlh52bFxCCCGaD6MrxA7Qbu+dC8C/x3TExaBjQ3I+Kw7mOjA4IZofKUqJS97
QdkG46HVstJw6WyplrWMDEkII0Ty1HwMGVyhIwqf4IC9f3xXQLslYd/Skg4MTQgjRbPSepf3d8g7kHSY6wJ3bBml
j4j7380EqLTYHBidE8yJFKXHJ83I1MiA+gDX2btqKpOUgp9UKIYQ4k8kL2o3Sbm99n0FtgpiWoJ1l+8/v98nMSkI
IITTtR0PbUWC3ws8Pgapy19B4Qr1dySis5L55u7HZ5feGECBFKSEA7RK+jfbOVOMCxelwUi7FEEIIUY+Ee7W/++Z
BfhKPjmpPXKAHOaVVPLlov2NjE0II0XyMelEbW+r4Btg7Dw+TgdlTeuBi0LHiYC7P/XzQ0REK0SxIUUoIYETHECy
KiU22DtqKo8sdG5AQQojmKbIXtL0KVDuseQF3FwOvTuqGToGFe07w+sqjMoitEEII8IuBIf/Ubq/4N5gL6R3rz6v
Xa1dnzNmYxicbUh0YoBDNgxSlhAACPU30jvFnlb2HtiJphWMDEkII0XwN+5f2d/98yD1Iz2g//jVaO6jxv1+TeHn
5ESlMCSGEgIR7IKg9mPPh12cAGNstnEdHtQfg2Z8Psmx/jiMjFMLhpCglxCk39Ytm9amilJq+BSqLHByREEKIZim
sG3QYB6iw5nkAbh3Uin+P0QpT76w5xn9+PiSFKSGEuNQZXGDMq9rtnZ9C5g4Abh/cipv6RqOqcN+83ezJKHZYiEI
4mhSlhDhlXLdwQqPbcsQeiaLaIPlXR4ckhBCiuRr2L0CBQ4sgex+gFaaeGd8JgI82pPLUogPYZSBbIYS4tMUOhG5
TABUW3w82K4qi8PS4TgxvH0y11c7tX+0mv8rRgQrhGFKUEuIURVF4alyn2rOlTu5a5OCIhBBCNFvBHaDzddrt1f+
tXT0tIZbnJ3RBUeCzzcf5v58OInUpIYS4xI18Flx9IScRtn8IgEGv460be9A5wpvCihpmH9CzJaXQsXEK4QBSlBL
idzpH+KBrdxUALmmrsNbI9N5CCCH+wNBHQdHB0aWQubN29Y19onl5ojb4+Tc7svj4iI71yflYbXawWeHAQvh2Gix
/HPKTHBe/EEKIpuERCCOe0m7/+iyseQFKT+BhMvDJ9MuI9nejyKIwdc4OHl+QSFmV/AYRlw4pSglxhuuvmUAJHvi
oZaxY+bOjwxFCCNFcBbaBrjdot1f/p85DE3tF8vrk7uh1CvuLdPz9sw3877n7KXyxM3w3HQ7+CJtnw+ze8OnVkPg

9WKsd0AghhBBNoud0iB0ENRXaeISvd4Z5NxF8chML7+jLgBA7AF9tTefK19ex5kiegwMWomkYHB2AEM2Nn5c7qaG
D8MlZxoltCygYPIoAT5OjwxJCCNEcDfknJH4Lx36F9C0Q3a/2ofGdg2ir+pC69E0GW9biqVaCBQpVT37WX84A30L
iijaipK2HtPXgHgDdp0C70RDRCwySe4QQwmnodHDzD9pYhNs/hvRNcHgxHF6Mn18czxnjcOkawaaUIoor7Bz5QsE
e6kOftlF4+gaBq492CaCrD7j5gWewdltRzvmyNruK1W7HZNA3TTuFuEhSlBKiHjEJE2DBMgbYd/HKiiM8P6Gro0M
SQgjRHPnHQfebYNdnsPQRbUDb/CQoSIKi43RQbXQ4tanZuzVLPa/lhRPdOWnWgRl6+Ezjxbg9tMlagFJ2Aja9pS0
GV4i8TDuqHjsQInufs0hVWlVDRqGZjEIzmUWV9GsVQOcIn6Z5D4QQQlwYgwt0magteYdgxyewdx5KUSpxpEI+XA+
//UrPP7X8EaM7eIWCV5j21yMIDCZUnQuZ5SqJOZXsy6lkV1UEx726E+XnTpT/qcXPjaHtggnykgMgwrGkKCVEPXR
trkBVdHTQpbNu+24S+8TQJVL+uRdCCFGPwQ/D3rmQvUdbfkd18STP1IqAMY/j3nYk1+l0jLXaWZKYzUvLDrO7BEb
uGUC/mNG8mJBDTNbPkLYBKvK0s6fS1ms7MrpDm5HQ6RpoM5Iyuwtvrz7G5mP5pBeaKTLXHX/E29XALw8OIdjbtUn
eAiGEEBcpuAOMfhkufxJr4g8k7/iVNq1bo9cBdhs5JRVsO3YSc3kJXlTgQwXeihlfpQJ/nRlPtRxqzFCYoi2/owB
Rp5bRACaYXzmQp0uns+O4R+12gZ4uzPtbP+KDvZqu3UKcQYpSQtTH3R8lsg9kbGGobg9PLornuzv6o9ed+/RYIYQ
QlyDfKBjzKhz+GfziIDAeAttCQBusrgFsWbqU0a0v1y7dAFwMOq7pEcGVnUJ5f90x3lt7jC3HSxia7sbk3g9w1y2
ziVazThWlNvxWpDq4EA4uxKp3Y6u9O+nVl5Fs70YFbgAEeLgQ5e/OybJqsooreXLRAd69uZcD3xghhBDnZfJE7XY
jR7J8aD10NHqjEYBQYBxQWGFh07F8Fiflsz4pn6ziSu1pWAhWigihiFCliBClCH+lFBesuFCDp95GrK+BWC87/lm
ruE6/gdGeyaxs8wSb6cbmY/mkFZi58cOtzPtbP1oHeTruPRCXNClKCfFH2o6EjC2MMOzhq/QR3PXVTt6Y3AM3F7k
eWwghxBl6TtOWM51jFlc3Fz33j2jLpN5RvLD0MIv2nmDe9gzmbc9gQHwAky8bxZXXzsCk18GJ3RTv/A7L3h8ItuU
wgs2McNmsvYR7MDr/VugDW4N/HJlKGOOWurB0fw7L9udwVefQxmq1EEKIRubv4cLVXcO5ums4qqqSVmBmR1ohWcW
VnCiuJLukioPFlfxaXIVNVRneLpjx3cMZ1j4YV+Op3y0Z22DBHbgVHmPcvrsZ13sWRbc+wY2fJXI4p4wpH27hm78
lEBvoce5ghGgEUpQS4o+0uRJ+fYbBhoN4Wa0sP5DLDR9u4aNpveXaayGEEA0m3NeNN2/swbSEGN5clcz6pJNsTC5
gY3IBvu5GJvSIxMPkwfvbBmOxDqCH4TiPxxymZ8U6dMVpGM15YM6DzC0ARAJrPYMYVfZv/u/H/SS0DsDHzejYRgo
hhPjLFEUhLtCDuHqKR6qqYlep/8qOqD5wx3r45SnY9gHs+Bi/Y6uYN3o21y/2JCmvXCtM3Z5AlL974zdEiN/ROTo
AIZqtkE7gHYneVsX8UVZ83Y3szSjm2nc2kpxX5ujohBBCOJnesf58PrMP6/85jL9f3oYwH1eKzTV8sjGVt1YlY7H
aGdQmiNfun0Hv295Cd/9eeCQNblsN130Mw/4N3aaATxRelpPMc30BtSyX55ccccnTTzquksoaP1qfw7Y4MR4cihBA
tkqIo5x5qxMVDG8Nq2o/gHQlFqfjOu4b5A9JpFeTBiZIqbvhgC5lF5qYLWgjkTCkh/piiaJfw7fiEtiWbWHDXM9w
yZxtpBWYmvLOJt2/sfv59WKshaycc3wh5hyG8u3YGVmCbs6ZvVVWVAydKWX4gh+UHcjhZVk2Ql0lbPE0Eemq3R3Y
KrffoiBBCCOcQ6efOg1e05b7L27Au6STfbMsgr6yKmQPjGNMlDOX3+cPNDyL8IKLnb+tKT8AnVxJZnM7nLs8zefs
TjOsWTv/4wKZvzHkUlFfzycZUPt90nLJqKwBGvcK1PSIdHJkQQjipVkPhrk3w4z1waBHeS+9hUd8HGbt/MKkFZqZ
8uJXZU3rQJcKnbr4RopFIUUqIc2lzpTZV69HlxA15hB/u6MttX+5h5/EiZn6+k9GRCj1KqogOMEBlEZTnQkkWZG6
D45sgcztYq37b3/7vYcW/wS8W2l6FPX4ke/QdWXqoiGUHcsgorKzz8kXmGo7mltdZ9+avSXw2sw+9Y/2b4A0QQgj
hKHqdwrB2wQxrF3xxT/QO146Ef3IVHcozmOPyMv+a78XCB65sNuMi5pRU8cG6FOZuS6eyxgZAoKeJ/PJq/vXDfjq
F+9A2RGaDEkKIRuHqA9d/BquegQ2v47n1NZa0S2Ws/UaSC82Mm72RaH93RnUJZUyXsGZToCox17AzvZBATxNdI30
dHY5oIFKUEuJc4gaDwRVKMuCVNvij8L2bH7nenhyvcseQbUN580EsumJc+IPBbD2CIKY/BHdEzdgKqRtQitJg63v
otr5HZ1VPgBrAMDWQXJdAjAGxRMS0ITAilnz8yLF6kWXxIK/CypaUAvZlljD9k218OrMPl0lhSgghRH38W8HUBah
zRtOrKonHy//LmysieOTqbg4Nq9Ji49UVR/h883EsNjsAXSN9uHtYPJe3D2bGnO1sSM7nzi93suiegXiY5F9VIYR
oFDodjHhKyxeLH8DtyAKWhGXyRNi/+DGpmvRCM++vTeH9tSlE+bsxuksYM/rHEubj1mQhniytYvvxIralFrI1tZD
DOaWop8bNeuemnlzZSSbycAaS6YU4Fxd3GPggbH0PKgsBFaWykFAKCa1nRLZi1YOT+JKqb8VhUxeS3btT6hGLp9W
IegJ2ZPSmzHwzA3WJDNPtYbh+NyFKMTFKHjHkaTspWgdFwB6IOr1jRQfuAdg9gpkbMoDHc4cw/ZNtfNZCClMllTX
oFPBylYF2hRCiyYR0Qrl5PtZPxzKYRCq3PsSmdl/Sv41j/onfmlLAP+fvI6BwD7fojhAQHMjArm3p0MoDxf0klNv
436TOjJm9hWMnK3j0h0TevKF7szg6L4QQTqvnNPCNgW+n4pK9nRd97+eZ8Q+zpSqW79Jc+fVIARmFlby/NoU5G9O
Y0ieau4a2JtjbtVHCqaqxsWLHYcJX30eH6n3Y7d3Js/Ulzd4dFVcCPV3IL7dw79e7+XB6b4a0DWqUOETTkaKUEOc
z9BFtsVm1wlRFPpjzsZbmsGvPPtr2vYK9Je78mqmwKrmUzKJTl+CZ0YpL5NfZnYvBnbLoqyiIv4ms1gEEeJVhKMu
C4gztjKySDO12WQ5U5Gmvp9qh4iS6ipPcxAHcgyt4IG+MdsbULX3oE9f8ClMllTUsP5DD4n3ZbEzOx9fNyNy/9ZP
LMYQQoilF9sYwZS7WLyZypX47pV/24KBHJ0I7Dca/3QCI7K1dxtGIKqqtvLTsMGu2bOVRw1xGmbZrD5QCG04tpwR
4BPNd15mM3tSOn/aeoE+sH1MTYhs1PiGEuOS1GgKzfoGvr4eiNEyL72EIMMTFE1tsV467tmN+fiTvZrfj001pzN2
WztR+MdwxtDWBng0zK/mxk+V8vTWdnTu28Lr9BeJ0uaDA1fqtXK3fik3vSk2rERi7XMs/9oax4EAxf/t8B5/N7EO
/VgENEoNwDClKCXGh9AbwDNYWQK2pITvNRI/W/RliNDKkLzytqmQVV5JfbqG8ykp5dQ1lVVbKq61UW+10ifChV4w
frsbfj+nhD/4xEPMHr2uzgrlAK1AdXQarnuPa0q+whhh5OHckM+ZsY86My+jbDL6MKy02VhzM4ae9J1h3NL/20gy
AggoLN3+0le/uSCAmQAZqF0KIJtNqKJYJn2BZcCfe9nI6mrfD9u2w/VVUFJTQLjDqRe1S8wa2KTmfZ+dv4rqyr1n
psgIXxYaq6FDajtI2qCwEc6H2t7IIKvKI2vEC2z18ecs8ktcXV9E10pduUb4NHpsQQojfCWoLt/4Km96CjG2QvRc
s5egzNtGKTTwM3Bbbn/ut97Im085HG1L5ams6UxNimJYQQ6Sf+zl3X1JZw8r9J9hwQiF1TQoWm0pljY2qGhspJyv
YmlrIcN0uvjC+jZeuklJTGOrI/+BTuBcOLERffBx90mJIWsxr7oHER93FyxntmfXpdr68tS89ov2a5n0SDU6KUkI
0IEVRiPRzP++X8kXRG8ArRFtCu4DeBVb+H9eXfIo11IXHcoYyY852/m9sR67tEXFGwatpqKrK0v05LP1xLh2qdhO
qBtBHDUUX1Jq+3bswuF0YD3+/l8M5ZUz5UCtMhfs23fXoQghxqXPvMg46jib14FY2rVmKe94ueipJxOjyIGcf9k/
HUjToaXyH3IVeX8/16eeQmFnCvG3H2Zuk44sT2yirslFSWUN1ZTnj7b/wteEH/AynJu2IH4Ey8jkI7nD2jmw1sO8

bWP8qroUpPGz8ltvVxcz/dAx+E++gzD2SQouewgoLhRUWzBYbCa0D6BHlK5f4CSFEQ/AIhCue1m7bbXDyCJzYBVm
7YO9cfHM2Mccrjd1j3+Dp3R7szSzhg3UpfLQ+heHtg5maEMug+EB0Ou072W5X2ZJSwLc7Mli6P4dqqx3Qw/HkM15
Y5R7Djzxo+A4dKmrMALwnfa7Fw3Uw4mmtSHZwIeyfj1Kczt3mZ+jpN5i7i6Yw/ZNtzPtbAh3DvWv3aLXZOV5oJi2
/gqoaO1a7nRqbitVmp8au4mbUc3n7YPw8XJrinRXnIEUpIVqaAfeB1QKrn+PG4g+whhl4Insgj/2QyIvLDjO5dxQ
394shyr8BC2PnkJZfwbM/7mZg6pu8ZVhe91ulDNjkAodiWRjWleeq2vJtcTtu/mgr39yeQJBXw5zuK4QQ4gLoDcR
1GUBclwFsTyvkoaWHOX48lSeMXzBOv5mAdY8zf80K5vj+negQP9qHejMgPpBukT4YzihUqarKuqR83l97jE3HCk6
t1eFTlMnlul1cqd/BYN0+3PQWAGyB7dFf+R9oM+Ic8Rmhx83Q9QY48AO2tS/jXXCUW2zfwTffAZCr+nJcDcGqBpN
vD+Oe5QNxD45lUu9Iru0RKXlFCCEaik4PIR21pcfN0Oc2+GYqSkESPX+dwsKRz7Fq+LV8simNjckF/HIoj18O5RE
b4M5NfWOorLHx3c6MOrOLtw32xMteSnxsFO4mI556G2HWLBJOzCEuZ7m2Ue9ZKKNe1HLCaYoC4d21ZehjsPYl2PA
6CZXrWO22l0eqb2HqxzqmJsRw7GQFSbllpJysqHPVRn1c9Dqu6BTCpN5RDIwPRK9rnAMcNTY7O48XoQB94vzlQMo
ZpCglREs05GGwVsH6V5ha9A6x3Xx5LP0yMosqeX9dCh+vT2JUWy/G9YjG4OpFhcWK2WKj0mLDbLER4OHC2G7hFzQ
1eLXVTn3f51U1Nt5dc4xla9fzmu5/dDIcB8DWYTx6mwUKj0FRGtgskH8U1/yjPAc84urO0uLLeOO94Tx8x234esm
lfEII0dQui/XnuzsSWH2kNQt3dSA34wtmmudwnW4t8cUZ3JH3AEsSA3ht5VG8XA0MaB3IwDaBDIgPZE9GEe+vTeF
wThmuVNNLn8GN4Xl0K1lN65rD6FRb7evYfWLQDXoAfY+p2pm/F0JvgK6T0HeeSMameZz85X/Eq+l4K2ZClGJClGL
6cAT0cI9hIW8XjOfVJWN4adkRhrUPZnLvKIa3D649Un/RrBbI2ALeEdqsVH/w48FssbI1tZD1R/Opsdn551XtZEI
PIYTzCu4Af1sNi+6FAwtQlj3K5Z22cfnUtziWb2bpph3sOXAQ76KTFC0vxKRYGasaUUwm2kUG0qNVCGE+7iTv+JW
2Vhu6nCNQcAxO5wydAUa/DL1nnjsOgwkufwLaj4Ef78Yn7yDvubzBYssW5q8ahIka2mGhm1KDp4uVAA8TezwHUWE
KxKDTYdQrGHQ60gvNHMwu5ed92fy8L5swH1cm9ork+l5RRAdcwMH95F9gz1wIaA3dbtDyxe8UlFezISWX1YfzWJd
0krIqKwBXdgrhxeu64usuZ2idJkUpIVqq4f8GWzVseotBR/7Leu9IarzLUSwVGKmB41CTpucj22het07EQt1/lF9
afphbB7Xi5n4xeJ4x5baqqmxOKeDLLcdZcSAXq93AE7t/xdvNiI+bEW83I1mFZhLKl/OD4VM8lGpsrv7oJ7yPvu3
I33Zkt0FJJhQka1/cBxbgVZbNJMNaJlWspfS1V6mK64fePw5dQCv0AXHgFwu+0WCUy/uEEKIxKYrC8PYhDG8fAvT
EnjwO2/e30K0qhVVe/8fXAfeyKUdHlaWaqkM2Vh+ysR4bMUoud+iO08V0nDglGx12OPm7HQd3gg5XQ/ur0YV2+cO
iznnpdEQNnEJE/xtRUKGqGIpStQMehamQ/Atu6Zv5h/E7bjat5/Gqm1h5sBcrD+bSJtiTey9vw5guYRd+5NtWA3v
nwrqXoThdW+cTBXFDoNVQ7BH9yKyA99elsimlkB1pRXWOwu/NLOazW/rIpSBCCOdl8oKJcyCqL6z4Nxz4AQ4vprX
Nwj2nt6nvKzDr1AK0B8j5/T69tSFKhv/74sY2jOgJf1sDa19C3fA6V+u3cLV+y9nbVQL2r2HEU9DrFtD9dubv/qw
Svt+ZyYLdWWSXVPHWqmRmr07myo6h3Da4Fb1i6hmnKvcArHgCjv3627q1L0J0Amq3G/nF3ofXEvWkb1mLqv62ib+
HC2VVNSw/kEti5nrevLEHvVvALOpNQYpSQrRUigJXPKv9E731PZTSzLNygFGxcafhJ0a5HuDDoEcp8ozHzWhga2o
BmUWVvLD0MO+uOcbMAXHM6B8LCszfmclXW49z7GRFnX1VWGxUWGzklJhprZzgH4Yfuda4EQA1bjD6az8A77C6Aej
04BejLfGXw8j/QPpmirfPw35gIf5qKaSsgJS6T7OiZ1vENLpNfQkPV/nnXgghmoIufhjcvha+uQm3nERm5TzDLKj
/B8bveQRjD+nMoaog2o5/CGNIu4aNS6cACrj7a0tEL+2BQQ/B/vmw4t+ElmXzscurJPn05/6SyRzIg7/P3c0bvxz
l3uHxjO0ajkGvQ1VVjheY2ZCcz4akfLamFuBlVLjDfyfjSr7A05yp7dvVF2rM2oy4e76EPV9iAobbo/juwFASbUO
w4E6ErxsD4gP45VAe+zJLmPzBZr6c1bfRpkoXQgiHUxTodyeE94DvZkBZtrbexVM7w9Q7XFsMrtoBdOupxWbBXlN
FRomdyJ4j0Id2gqAO2rZ/9uDFqbOmlPZjYNWz2sQZRjdtvcFVWwqPQU4i/PwgJH4HY9/UBnUHOkf40DnCh0dHtee
XQ7l8sz2D9Un5LDuQw7IDOfSK8eO2QXFc0TEUfUUurP4P7P5Smxldd+qS8+LjkLIG0jejpG9mmGqkxt6bH5X+5Ic
OZHCHCIa1D6ZbpC8HT5Ry79xdpBWYmfzBFh4Y0YY7h8Y32mWDLYUUpYRoyRRFmzGp1y1QUwFGD3Bx15KC0R2SV8J
P9xFrTuE/efdC1yeh753UqPDjnhO8szqZlPwKXv/lKHPWH8Vgr8JcA3YUfF2MjO0eyaQeoWSv/4qewTaMJ7bjnrs
DY3URAKqiRxn2L5SBD2gFqPPR6SB2AL6xA9jf7yme/PJrvMtTiFZyiVbyiFHyiFZy8VSq6J81h1UvHqJ89Gyu7hX
/5y/D+BOsNjtpBRUczC7Dy2RgaLsgufZbCHFp8IuBmStg5RNwbLV2OYXOoF1SpzNgV/QoXmEoYV0hrJt2dNsrFFt
NDclLltD2jMsXGpWiQJeJ0PYq7eymzW/TpmQTP7MJi4cH2TZvcku8OfmDD4t+DsTLN5DjxRYKK1Ws6IhAzyRsTLa
uplW2dtj+pOrNV/prORoyieST5YQU72aAbj8DdPvppByngy6D/9N9wWOu32PuMAnvwXehBLcnOa+Mmz7aytHccq5
/XytMNdXYjkII4RDR/eC+vdqZpZ4h4Op93qfYamrYs2QJ4X1Hozc24OXOET1h6oL6H7PbYNsH8OuzkL4Z3hugHdQ
Y+IBWvAJcjXqu7hTM1W3cSM30YPHmRPYeTcUro5Ttc8sodytmvP1XjPZT42N1HK+deeXfCrtd5cf1O0hdNYer1TW
01WUxTr+ZcfrNUPkxVI0D60RgIF0ifVj890H8e0EiC/ec4JUVR9mYXMCz13Qi0s/dIRNWNQdSlBLCGQS3r399h7E
Q2Ue79jtpOSz/FxxZinHEU0x0O8GEXgfJSd5NTfYBImxZGPR2+P134T5t6QJ1z2YyukNUH5Sh/4Lovn8q5M5RAbz
12L3Y7SrVVjvVVhtVNXYKaqwc2/IFHXb8m+HqFvb9dD23bnmG+64Z3DBTgtdUakc2fje2SV5ZNeuyFTYsPMCR3HK
O5JSdmh1Ec033cF6c2BWT4dJMFEKIS4yLO4x5td6HLm5eviZi8tRmi+pxMyx7FJJ/wcVWQQwVxOhOHcG3Avmntq/
nd5DZ4MN8t4m8UjiIkmoXOFICwDFdNyojhlAZH0hVuIph8zt0r96C8eRhfPZ/Bvs/g1bDiO9zG9/fNpCb5uzieIG
Z69/bzJe39iU+2LNJ3gIhhHAIgwkC2zg6inPT6bUzu9qPgZ8fgqQVsOZ52PmZlu+qy7SlxgxAHHAv1K2UaMNBscs
ez9vGW/BhIEPSTcRWFPPfJYfYmloIjGFt5BReHWBD2TSbVuY9KBV5sOtzbfEMgbZX4ukXx+sdIxnvb+LZ9WVsS7E
x4jVtwhAPFz2BXiYCPFwI9DQR5GUixNuVYC8Twd4mgr1cCfNxJcDTuSb1kKKUEM7OKwSmfAM758DyxyFtPXx0OaD
9uAg/vd05TgSqNnhhbDUIXWx/iO4PYV3rzojxF+h0Cm4u+jqDrsdcfSfVHbpQOXcKXa2p/Cf/Pm595x906DmQB65
oS4TvRY43ZbdDymrY/hEcXaadcuvmj809kEyLBwdLTHSy+7I1vQMp9k5U4467i542IV4cyCph4Z4TnCiu4v2pvWS
sECGEaK4C28DN86GqFMrzoDwXynOpLsnhcFIy1soSQj0NBHvoMSp27fJ3uxXCe+B+2Symmry4vsbGnoxi9meVEBv
gQd9W/rWDl9fU1LAkeTidR72MMXOzduT9yBItv6SsJsrNj2VtxvLv5PYsKIxm8vub+WTGZQ1zQEUIIcRf4xsNU77

VxsFa+giUnah/O6M7uAdol4u7+WN19Se53IVfKtvydk4HKivssCuLH3Zl1T7FzajnH1e2Y0b/WOw2K0vSi4m+6gu
MWVth//dwcJGWk3Z9Dmg/u4YBw/Rg1+tIVsPZaOvEJmsnthZ04HjBuSeC6t86gNsGtWJI26AmvZqksUhRSohLgaJ
oM1nEDYGf7oMTe7R/3oM7ajNpnF7cA7RTXFW7NhOG3UZNjYVlq7cweswYdA15mu15mFoPhLvWYP1yEmGFSXzn8jQ
P7L6LoXsuY/Jl0dw9LJ4wn/MUp6pKYM/XWjGqILnuY5WF6CsLiQFidIAOZrIMu2LAEtYLl3Yj0bUZwaaSaF6ctwL
v9J3M/d+33NzWhrc5Xdv3qfcIu1X7C9DpWhjyyIXPMiWEEKJhuXprS2A8ACag24ALfKpRT79WAfRrFfDHGykKtBq
iLUXHtRyz7xsoz8V93+e8BjzmHsgP1X15+p391AR3ZVjnKEZ2DKFTuLdcDi6EEI6iKND5OogfAZk7tPGnTF6nFm9
tCBRD3QPQBrTB2dsDt1lt7EwrYm3SSdYdzedITikD2wTxn2s6116yffonATr9b7li9KtwbBVkboOSLG0iqJIMKM1
CZ7fSVsmkrSGTW1iOqugo8e1Epm8fEj0T2Ku2Ja/cQm5pFXll1eSXV7PpWAGbjhUQH+zJrQPjuKZHRIu+9E9+NQl
xKQloDTMWX9xzamr+/OCDf5V/Kwy3/QLfzcA9ZTXvu7xOnurLnp2t+XpnGwLa9Wf0VWMIDghArThJUeZRctOPUJa
djL4gic5l6zGpVQBU6z04EjqW7YHX8PneclyqCwlUSujmV8M1bYy4ZG4lzp6KrvAYrie2womtsPpZ+gM/KmgD/Vq
A/eeJed1LkLYBJn5y9sDvQgghnItfDIx8VhtbJG29NojuwZ8Iqs7ndsPP3G74mepiI3vXt2Ld2nZ85taZwI6DaRc
bTZS/G5F+7gR5mpziSLcQQrQYrj7aJEwXyWTQ0z8+kP7xgTw2ShuH1qC/gAvbDS7Q7ipt+T27DcpyIGsHpKyF1LU
oBcn4FiXiW5RIZz7mRt8YbfzELtdDcAeyiiv5dGMq87ZlkJxXzqM/JPLy8iNMviyKzhE+tAryIDbAo0UVqaQoJYR
o3tx84abvtGlnt39EsL2YkfqdjGQnJM/D/tZ9VOlccFWr8QfOnFj1iD2Sz20jWVA1EPMxVzhmB9xpExzM9JFtubJ
TKFarlSVLlhA1ejTGskxtitfkVZC6Fizl4OJFjW8cW4t92GP2J5MwhvZoR9twfyL9PXFxcQFFr01VvvRRSN8E7w+
C6z6CVkOb/C0DUFVVjsYLIURT0em17/tWQ7Uj4skrYf987KnrMZnz6aMcoY/uCNQsgr3/Zd2uLrxiG8dme0dc9Ho
i/NyI8nfnshg/BrQJpGuEz4X90BFCCOEwf/l7WqcHnwht6TheW1eSBanrtN8jR5Zqs/utf1VbQjoT0WUijw+5mb9
f3oZvtmcwZ2MaWcWVvLPmWO1uFQXCfdxoFeRBhzBvEloH0CfWHw/TxZd/Ki02MovMRPq51xlupSFJUUoI0fzpjdo
sgyOeguy9qJnbKTi8CTVzB0H2PFzVauyqQg7+5BnCqPSIRPGLpTCwN6ke3fGx2JhksVFRbcViszOsXTBju4XXP/2
qfxz43wqX3QpWi1aUcvPDqCj0stj44pvdLD+Qy7ztp0LTWWgdZKRjmDsdwxOIHfQN/XY8gHfJEdTPr+Fk7wep7Pc
Aob7ujT5QellVDYv2nmDutnQOZZeR0CqAMV3DuLJTKP4yFpYQQjQNo6s20UiHsehUFQqOQcYWrGmbqE7ZhEdZKoP
1iQzWJ7LbHs+71rGszO9Fan4F646e5NWVR/EyGejXOoCB8YF0jvChqsZGWVUNpZVWSqtqKK2sodpmx6jTYdArGPU
6DDrtb1ygB5fF+eN5nh8fqqpSY1NxMfzuR5XdDuaiRn6DhBBC/CGfCOh+o7ZYzHB0KSR+D0krIXe/tqx+Hq9uk7m
1313M6D+UpftzWH04j5T8ClJOllNaZSWruJKs4krWJ+XzwboUjHqFHlF+DIgPZEB8AO1CvfA0Gc46iK2qKoeyy1i
fdJJ1SSfZnlqExWZHUSDKz502wZ7Eh3gSarI2WJOlKCWEaDmMbhDdDyW6H4H970VVVXYfOkruyZNExrYlPjyA8IY
8VdXgAobfzr1yc9Hz7k29mLMpjTVH8jh4opSCCgtHc8s5mlvOwj3agIkmHuMpw2fcaFhN8I5XWbd1BXfZbqTIsx3
RgR7E+HsQHeBOkJeJ6hoblTU2zBYblRbtr8mgI9zXjXBfNyJ83Yjwc8PP3agljeoyKD0BpVlQegK1LIfschtbM6v
ZkllNkdWID670VI3oUqpZk1LNjkUWugQZaOXZcMlDCCHEBVAUbWyrwHgMPW7W/vEuSoNNs2H3F/SwJvOBy+uYfdq
wO2oaP5Z3YPlxlZLKGlYezGXlwdzzvoQOO+FKPq2UbGKVHAoxs19nI8JLR7S3gQgvPYHuCpXVNZSYqyk2Wyg1V1N
SaUG1VhOsryBIV4Y/pXjYyzBWS64QQohmwcVdGwOr83VgLoRDi7TB0rN21s7qZ4gfwdiEuxk7aRgoCqqqUlhhITW
/gpSTFexKL2J9Uj5ZxZVsSytkW1ohr/9yavcGHYEeLgR76GnnWkiEmsvJ3BPoq4vxVcoZQQXX6SrwMVRit4NaBpS
BekyhvNp2ztAvhhSlhBAtlqIo9OjYDmjXZK+p0ynMGhjHrIFxqKpKXlk1B0+UcuBECYdyyigx12C2WPnM8iBp5i4
8YHmv9oh4alUIS9L7siS1H9+oMZxzykPAmwr66A7TX3eAvvrDxCh5eGKus42CNoPitcC1OrSxr+pTBKU56l9/A4Q
QQvw1frEw5hUY8k/Y8i5s/wj3kiQGlDzBAOBFzxDKItqTpMSysTycfeU+BBmrCTaaCdSbCdBV4KuU41eTh3/Vcfy
rszCqlrNfx3xqydHuGgFvIOr325w+jmNvtNYKIYRoCO7+0GsG9JwOGdtg81twaDEk/6ItfrHg3xrFM4QArxACPEP
p7RnMpC4eqO0rKSgpI/lEIWm5BZzIL8KvJpdYJYfYyhyiqk5iUH6XCOqb2+qM4/6l+ob7XSFFKSGE+JMURSHE25U
Qb1eGtQ+uZ4vBkDcFdfV/IWkFcdZc7tYt4m7DIk4aI9htuowaoyeKwXRqcUVvNOFhziC8aAdR1UfR1/NLoUR1J1s
NIEf1J0/1xUVnp5UPxHiqeOstKJYKsFZpR1eMHpgxkW3WsT/PAixt9PdFCCHEBfAMhhFPwsD7YfvHsHcu5CehlOf
iXZ5LL9bS6/S2NefZl94F/FtrE5q4+1Nm1XOi3E5miZXU4hoKq0Cn0xPo7UqItxvBPm6Eervh6+lGieJNns2TLIs
Hx6vcOJBbA4xo1KYLIYT4kxQFovtqS2EKbH0fdn2hnYVblFb/U4DAU0u/0yvPqARZ9W6UukWi8wzGyz8Ivbs/uPl
pi4un9rqqCqigqthKy+CFBxqkSVKUEkKIxhTcAWXyF1BdDkeXwcGFkLSSoJosRtZknf/5AfHYYgZSGNSXYp/2mE3
BVCluWGx2bFY7fipcFuuHr/sfjxnlDrQGfAsK4LnAhmqZEEKIhuDqA4Me1BZLBeQehJx92rghOYnazEyuvtrEH6d
/ILj5gVcoBLTRLg/0idIGzD3FC+0c4nZo44Pkl1vwczfWOyivB9oZt91P3S8oKOCdvzVym4UQQvx1/q20cXeHPqb
N4FeWC+W/W8pyocasDYGidwGDKxhM2uIVeupgRjwEtMbgFYb/RUySZC8oAKQoJYQQLYfJ89R0rhN/K1Bl7wFr9W+
LrVobXN3dD2IHaYtPBHog6NQihBDCibl4QNRl2tJAFEUhyMvUYPsTQgjRzLj5QnzLPcPVoXPNvvPOO8TFxeHq6kq
vXr1Yv369I8MRQoimcbpANfI5GP0yjHsTJrwP138KN34N49+Gbjdos28IyRVCCCHOS3KFEEK0TA4rSn3zzTfcf//
9PP744+zevZtBgwYxatQo0tPTHRWSEEKIZkZyhRBCiPORXCGEEC2Xw4pSr732GrNmzeLWW2+lQ4cOvFHGG0RFRfH
uu+86KiQhhBDNjOQKIYQQ5yO5QgghhBiuQKIYQQ5yO5QgghBiWi6HFKUsFgs7d+5k5MiRddaPHDmSb775
2hwx0np+fj81mIyQkpM76kJAQcnJyztq+urqa6urq2vslJSUAFBYWUlNzvjlym5eamhrMZjMFBQUYjUZHh9MoLoU
2grTTmVwKbQTtOxO0mZhaAskVzv2ZvBTaCNJOZ3IptBEkV7Qkl8Jn8lJoI0g7ncml0EZo2Fzh0Nn3lDOmHFRV9ax
1AM8//zxPP/30Wevj4uIaLTYhhHBWBQUF+Pj4ODqMCya5Qgghmp7kCiGEEOcjuUIIIYQYrkCiGEEOcjuUIIIQ
KAfDYY4/x4IMP1t632+0UFhYSEBQb7JpzkpLS4mKiiIjIwNvb29MoLoU2grTTmVwKbQTtaHB0dDT+/v6ODuW
CSK5w7s/kpdBGkHY6k0uhjSC5oiW5FD6Tl0IbQdrpTC6FNkLD5gqHFKVcXFzo1asXK1eu5Nprr1eu5Nprr61eu
W9iaTCZPJVGedr69vY4fZqLy9vZ36QwqXRhtB2ulMLoU2Auh0Dpvj4qJIrrg0PpOXQhtB2ulMLoU2guSKluRS+Ex

eCm0EaaczuRTaCA2TKxx2+d6DDz7I1KlT6d27NwkJCXzwwQekp6dzxx13OCokIYQQzYzkCiGEEOcjuUIIIVouhxW
lJk+eTEFBAc888wzZ2dl07tyZJUuWEBMT46iQhBBCNDOSK4QQQpyP5AohhGi5HDrQ+V133cVdd93lyBCanMlk4sk
nnzzrtGFncim0EaSdzuRSaCO03HZKrnBOl0IbQdrpTC6FNkLLbafkCud0KbQRpJ3O5FJoIzRsOxW1pcz3KoQQQgg
hhBBCCCGcRssYwVAIIYQQQgghhBBCOBUpSgkhhBBCCCGEEEKIJidFKSGEEEIIIYQQQgjR5KQo1QSeeuopFEWps4S
Ghjo6rL9s3bp1jB07lvDwcBRFYeHChXUeV1WVp556ivDwcNzc3Bg6dCgHDhxwTLB/wfnaOWPGjLP6t1+/fo4J9k9
6/vnnueyyy/Dy8iI4OJhrrrmGI0eO1NnGGfrzQtrZ0vvz3XffpWvXrnh7e+Pt7U1CQgJLly6tfdwZ+tFZSa5o2Z9
JyRUaZ+hPyRXO0Y/OSnJFy/5MSq7QOEN/Sq5ouH6UolQT6dSpE9nZ2bVLYmKio0P6yyoqKujWrRuzZ8+u9/GXXnq
J1157jdmzZ7N9+3ZCQ0O54oorKCsra+JI/5rztRPgqquqtO/S5YsacII/7q1a9dy9913s2XLFlauXInVamXkyJF
UVFTUbuMM/Xkh7YSW3Z+RkZG88MIL7Nixgx07djB8+HDGjx9fmyCcoR+dmeSKlvuZlFyhcYb+lFzhHP3ozCRXtNz
PpOQKjTP0p+SKBuxHVTS6J598Uu3WrZujw2hUgLpgwYLa+3a7XQ0NDVVfeOGF2nVVVVVWqj4+P+t577zkgwoZxZjt
VVVWnT5+ujh8/3iHxNJa8vDwVUNeuXauqqvP255ntVFXn7E8/Pz/1o48+ctp+dBaSKzTO8JmUXOFc/Sm5wjn60Vl
IrtA4w2dScoVz9afkij/fj3KmVBNJSkoiPDycuLg4brjhBlJSUhwdUqNKTU0lJyeHkSNH1q4zmUwMGTKETZs2OTC
yxrFmzRqCg4Np27Ytt912G3l5eY4O6S8pKSkBwN/fH3De/jjyznac5S3/abDbmzZtHRUUFCQkJTtuPzkRyhXN/Jp3
lu+U0yRXO0Z+SK1oeyRXO/Zl0lu+W0yRXOEd/NmaukKJUE+jbty+ff/45cv58MPPyQnJ4f+/ftTUFDg6NAaTU5
ODgAhISF11oeEhNQ+5ixGjRrFV199xapVq3j11VfZvn07w4cPcPp7q62tGh/SmqqvLggw8ycOBAOnfuDDhnf9bXTnC
O/kxMTMTT0xOTycQdd9zBggUL6Nixo1P2ozORXPEbZ/xMOsN3y+9Jrmj5/Sm5omWSXPEbZ/xMOsN3y+9Jrmj5/dk
UucLQYNGKPzRq1Kja2126dCEhIYHWrVvz2Wef8eCDDzowssanKEqd+6qqnrWupZs8eXLt7c6dO9O7d29iYmL4+ee
fmTBhggMj+3Puuece9u3bx4YNG856zJn684/a6Qz92a5dO/bs2UNxcTHz589n+vTprF27tvZxZxZ+pHZyK54jfO+Jl
0hu+W35Nc0fL7U3JFyyS54jfO+Jl0hu+W35Nc0fL7sylyhZwp5QAeHh506dKFpKQkR4fSaE7PAnJmlTQvL++saqq
zCQsLIyYmpkX277333suiRYtYvXo1kZGRteudrT//qJ31aYn96eLiQnx8PL179+b555+nW7du/O9//9//3O6fnR2kiu
c+zPZEr9bTpNccbaW2J+SK5yD5Arn/ky2xO+W0yRXnK0l9mdT5AopSjlAdXU1hw4dIiwszNGhNJq4uDhCQ0NZuXJ
l7TqLxcLatWvp37+/AyNrfAUFBWRkZLSo/lVVlXvuuYcffviBVatWERcXV+dxZ+nP87WzPi2xP8+kqirV1dVO04+
XCskVzv2ZbInfLZIr/lhL7M8zSa5omSRXOPdnsiV+t0iu+GMtsT/P1Ci54s+Oui4u3EMPPaSuWbNGTUlJUbds2aJ
effXVqpeXl5qWlubo0P6SsrIydffu3eru3btVH3ttdfU3q8ePH1dVVVVfeOEF1cfHR/3hhx/UxMRE9cYbb1T
DwsLU0tJSB0d+cc7VzrKyMvWhhx5SN23apKampqqrV69WExIS1IiIiBbVzjvvvFP18fFR16xZo2ZnZ9cuZrO5dht
n6M/ztdMZ+vOxxx5T161bp6ampqqr79u1T//Wvf6k6nU5dsWKFqrO0Y/OSnJFy/5MSq7QOEN/Sq5wjn50VpIrWvZ
nUnKFxhn6U3JFw/WjFKWawOTJk9WwsDDBVaDSq4eHh6oJE9QDBw44HeUmb1Spw1jJ9+nRVVbXpPp988kk1NDR
UNZlM6uDBg9XExETHBv0nnKudZrNZHTlypBoUFKajUY1OjpanT59upqenu7osC9Kfe0D1Dlz5tRu4wz9eb52OkN
/zpw5U42JiVFdXFzUoKAg9fLLL69NHKrqHP3orCRXtOzPpOQKjTP0p+QK5+hHZyW5omV/JiVXaJyhPyVXNFw/Kqq
qqhd3bpUQQgghhBBCCCGEEH+NjCklhBBCCCGEEEIIIZqcFKWEEEEIIIYQQQgghRJOTopQQQgghhBBCCCGEaHJSlBJ
CCCGEEEIIIYQQTU6KUkIIIYQQQgghhBCiyUlRSgghhBBCCCGEEEI0OSlKCSGEEEIIIYQQQogmJ0UpIYQQQgghhBB
CCNHkpCglRAtlsViIj49n48aNDbrfxYsX06NHD+x2e4PuVwghRNOTXCGEEOJ8JFcIR5KilGgWZsyYGaIoZy3Jycm
ODq3Z+uCDD4iJiWHAgAG16xRFYeHChWdtO2PGDK655poL2u/o/VV1+Noih8/fXXDRSpEEI0DMkVF09yhRDiUiO54uJ
JrhCOJEUp0WxcddVVZGdn11ni4uLO2s5isTgbbnfe4tZbb22Ufd9yyy289dZbZjjbvJvIYT4KyRXXBzJFUKIS5H
kiosjuUI4khSlRLNhMpkIDQ2ts+j1eoYOHco999zDgw8+SGBgIFdccQUABw8eZPTo0Xh6ehISEsLUqVPJz8+v3V9
FRQXTpk3D09OTsLAwXn31VYYOHcr9999fu019RwB8fX359NNPa+9nZWUxefJk/Pz8CAgIYPz48aSlpdU+fvpowSu
vvEJYWBgBAQHcfffd1NTU1G5TXV3NP//5T6KiojCZTLRp04aPP/4YVVwJj4/nlVdeqRPD/v370el0HDt2rN73ate
uXSQnJzNmzJiLfJchLS2t3qNHQ4cOrd1m3LhxbNu2jZSUlIvevxBCCNCbJFb+RXCGEEPWTXPEbyRWiuzZOilGgRPvv
sMwwGAxs3buT9998nOzubIUOG0L17d3bs2MGyZcvIzc1l0qRqRJtc95+OGHWb16NQsWLGDFihWTbOGnTt3XtTrms1
mhg0bhqenJ+vWrWPDhg14enpy1VVX1Tmysnr1ao4dO8bq1av57LPP+PTTT+skoGnTpjFv3jzefPNNDh06xHvvvvYe
npyeKojBz5kzmzJlT53U/+eQTBg0aROvWreuNa926dbRt2xZvb++Lag9AVFRUnaNGu3fvJiAggMGDB9duExMQ3B
wMOvXr7/o/QshhKNIrqhLcoUQQpxNckVdkiuEw6lCNAPTp09X9Xq96uHhUbtMnDhRVVVVHTJkiNq9e/c62z/xxBP
qyJEj66zLyMhQAfXIkSNqWVmZ6uLios6bN6/28YKCAtXNzU297777atcB6oIFC+rsx8fHR50zZ46qqqr68ccfq+3
atVPtdnvt49XV1aqbm5u6fPny2thjYmJUq9Vau83111+vTp48WVVVVT1y5IgKqCtXrqy37SdOnFD1er26detWVVV
V1WKxqEFBQeqnn376h+/XfffdpwvfPvys9YDq6upa53308PBQDQaDOn78+LO2r6ysVPq1mq/NYjx4
91KeeeuoPYxBCiKYmuUJyhRBCnI/kCskVwvHkmlKiWRg2bBjvvvtu7X0PD4/a2/JyyidqgmCgDO3fuzBkzhs8++4zJkyfXvvbmm29m0qRJvP766+h0OubMmcM111yDt7f3Wa+zWq1MmjSJBx98kJEjR573+7z/3jMbvtLWTkk1ab4vK6e
7OXbsGJWVlVlFgISEhHzh+x0P4xuePYN8Ds8xXJs4c/+c5Zu9/nVYnnnmt0p+vkt2zpuGLK/ZQ+C8NXw7u+/yS7x5RlGJi77/cPXRE3rC6vn8vqqX30qcq98z2uqLs34hjtfVNiN7PUSHKiFTebd49Z/4+c+7kiDdDjBHTt25TrMs6CTdPOGBzJqJLu+3I6lJWvD7EbOTrlv59RrWoXPYZ6tjurCwdEH3DKcCH4ZS4IupZqmt4DUXL/s0bX1M1b6xIUSVuXjRlJS+uTcCf7fKtVa8u0nmFPd5WVWgnNNsO09oC9ZCCFEVsxN
zZMiQel8+Wdh8FW72Ouaz8fFk6KF4xaVD+4i2+Lb8+7Y5jV78V3ahS7Ozlz7V9SZ7J0eoc3NibTt5Mid1ea1v4Dc2y2Y+pMcfBYsHwpqe9f0u3yvaK8W1xtA7U9X8Tkis6vRSxdyOkX9bK22mwqljpR2aq
tw8LCSEpKuqDXVMn9zeQR+6HqeY+K5ySvuLoJ4fUXTBbQDoS7YuTiTUKFo5PGMmcYQF8fpl1C6VGgN5phJ+q+uPCR0tDhuxS9O3eF6/O60M7vOB4Bk7FGPfBYZxUnj0JqFmsX5Op5i4o9x0c2NZAfWF3
z5jB58mTc3/cPvAwMDa5HSm0NDQs95HLy8UoU8wfRbwKvnUcnY3TwkhhnbLwH1v8O2tJvxwD8z3k+YeHsYohYGzV4xqOI+cq7MhmmFMK5pH7FQeYWSoAzx
IrpBcIVoOKUqHQ2Ylqz57Mnz+f2NhYDIaXNC7vy4sKoQO6sthu+kw+DxDt3Tw/E7fJRH3mx3JCpU/cnUzJv8Ju7IgDeiz6AaWmEZQ5yJ5v
s2ZNvvvmG4ODDhP3WdNUCXLl2w+2w+2sXbv2rCMpPkdyPdg2ePRoPDw+dunTh3G0h2vW7/17gw3u0eQkV5+DvQ8VYw5+9/eBO0s+BH6hLr/
nz+eZZ55h5nh6dKDKL9V5ffvIITY8ePSS5630II0VxIUlJPcBIYycPHx/f+ZCtX7uyLlSEo7nNSHfFeXqXceOnj9xMxxjyVzbSgXtS5DLr7B

zZ2Kz2fD09GTWrFk8/PDD/Prrr+zfv58ZM2acdSrp8OHDmT17Nrt27WLHjh3ccccddY5O3HTTTQQGBjJ+/HjWr19
Pamoqa9eu5b777iMzM/OCYo2NjWX69OnMnDmThQsXkpqaypo1a/j2229rt9Hr9cyYMYPHHnuM+Pj4OqcH12fYsGF
UVFRw4MCBi3jXNPv372fatGk88sgjdOrUiZycHHJycigsLKzdZsuWLZhMpvPGIYQQzZnkCskVQghxPpIrJFcIx5K
ilGiRwsPD2bhxIzabjSuvvJLOnTtz33334ePjU5sgXn75ZQYPHsy4ceMYMWIEAwcOpFevXnXnX28+qrrrxIVFcXgwYO
ZMmUK//jHP+qc3uru7s66deuIjo5sWLZhMpvPGIYQQzZnkCskVQghxPpIrJFcIx5K
ilGiRwsPD2bhxIzabjSuvvJLOnTtz33334ePjU5sgXn75ZQYPHsy4ceMYMWIEAwcOpFevXnX28+qrrxIVFcXgwYO
ZMmUK//jHP+qc3uru7s66deuIjo5mwoQJdOjQgZkzZ1JZWXlRRzjeffddJk6cyF133UX79u257bbbqKioqLPNrFm
zsFgszJw587z7CwgIYMKECfWe/ns+O3bbswGw289xzzxEWFla7TJgwbuXPnctNNN53zVF8hhGjuJFdIrhBCiPO
RXCG5Qjiwop554asQTmzo0KF0796dN954dN954w9GhnGXjxo0MHTqqUzMxMQkJCzrt9YmMqIiI0aMqHfAxL/i5MmTtG
7dhAXF9dg+xVCiJZcs4QQlzqJFecn+QKcShHKkcShHKy6uprk5GSeeOIJJk2a2adEGJA7Rryl966SXS0tIaNJ7
U1FTeeecdSRxCCNGSK4QQghxPpIrREskA50L4WBz585l1qxZdO/enS+++OcrhBDOnz706dOdnz+vQGj6dPnz706dOnwfcrhBD
iz5NcIYQQ4nwkV4iWSC7fE0IIIIIYQQQgghhBBNTi7fE0IIIYQQQgghhBBNTopSQgghhBBCCCGEEKLJSVFKCGEEKLJSVFKCCGEEEI
IIYQQQjQ5KUoJIYQQQgghhBBCiCYnRSkhhBBCCCGEEEII0eSkKCWEEEIIIYQQQgghmpwUpYQQQgghhBBCCCFEk5O
ilBBCCCGEEEIIYRoclKUEkIIIYQQQgghhBBN7v8B/N0ecz4HNLMAAAAASUVORK5CYII=",

```
"text/plain": [
"<Figure size 1200x500 with 3 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plot_psd(\n",
"    trials_PSD,\n",
"    freqs,\n",
"    [channel_names.index(ch) for ch in ['C3', 'Cz', 'C4']],\n",
"    chan_lab=['left', 'center', 'right'],\n",
"    maxy=500\n",
")"
]
},
{
"cell_type": "code",
"execution_count": 20,
"metadata": {},
"outputs": [],
"source": [
"import scipy.signal \n",
"\n",
"def bandpass(trials, lo, hi, sample_rate):\n",
"    '''\n",
"    Designs and applies a bandpass filter to the signal.\n",
"    \n",
"    Parameters\n",
"    ----------\n",
"    trials : 3d-array (channels x samples x trials)\n",
"        The EEGsignal\n",
"    lo : float\n",
"        Lower frequency bound (in Hz)\n",
"    hi : float\n",
"        Upper frequency bound (in Hz)\n",
"    sample_rate : float\n",
"        Sample rate of the signal (in Hz)\n",
"    \n",
"    Returns\n",
```

"   -------\n",
"   trials_filt : 3d-array (channels x samples x trials)\n",
"   The bandpassed signal\n",
"   '''\n",
"\n",
"   # The iirfilter() function takes the filter order: higher numbers mean a sharper frequency cutoff,\n",
"   # but the resulting signal might be shifted in time, lower numbers mean a soft frequency cutoff,\n",
"   # but the resulting signal less distorted in time. It also takes the lower and upper frequency bounds\n",
"   # to pass, divided by the niquist frequency, which is the sample rate divided by 2:\n",
"   a, b = scipy.signal.iirfilter(6, [lo/(sample_rate/2.0), hi/(sample_rate/2.0)])\n",
"\n",
"   # Applying the filter to each trial\n",
"   ntrials = trials.shape[2]\n",
"   trials_filt = np.zeros((nchannels, 200, ntrials))\n",
"   for i in range(ntrials):\n",
"   trials_filt[:,:,i] = scipy.signal.filtfilt(a, b, trials[:,:,i], axis=1)\n",
"   \n",
"   return trials_filt"
]
},
{
"cell_type": "code",
"execution_count": 21,
"metadata": {},
"outputs": [],
"source": [
"# Apply the function\n",
"trials_filt = {1: bandpass(trialscl1, 8, 15, sample_rate),\n",
"   2: bandpass(trialscl2, 8, 15, sample_rate)}"
]
},
{
"cell_type": "code",
"execution_count": 22,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABKUAAAHqCAYAAADVi/1VAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAACQ0UlEQVR
4nOzdeXxcdb3/8deZyb4nTZu0dKWUtWWRfS9bEUQRBDEC26Xe1W8CIiiP+8tXgXlXoErCNerXFC5CG4oKiAFppVD
K1rKXrdB937Kvk8z8/jjTtKFLkjaZSTKv5+PRx8ycOTPz+UrMN/M+3yVIJBIJJEmSJJKmSJEmSpBSKpLsASZIkSZIkZR5
DKUmSJEmSJKWcoZQkSZIkSZJSzlBKkiRJkiRJKWcoJUmSJEmSpJQzlJIkSZIkSVLKGUpJkiRJkiRJkiQp5QylJEmSJEm
SlHKGUpIkSZIkSUo5QympF+655x6x6CIGDJkiV9fu0DDzzAAQccQH5+PkEQ8Morr3DHHXdwzz339HudkqqSh6+GHH2b
mzJnpLkOSNIBmzpxJEAS79NrN30nmzZvX47+39BQYSglDaD169AcD+fk6Poo4/y4n7LPPPPsvfee9tJSJK28fD
DD3P99Qc9M0oEd6WOWEo2wTUbiaZtHPWZGaLmnY/hi5LEVJ6x8R/U3z/7Rx9XTzzA4XaSzn4rMrEu93b+nPmV4NKy8qPo8aUDdrQ2yp1r0B9HHa2Xg37z5rT+jgNDb8fswQ9j45+4EGKS8P+RDZ4fBwVmqKtB5ALZ9gdR0Drjk2gW2ie+pLbqFVSczpuNVhE5QrUO0xQ6jLDW8MOhi4yUoaMmhH/UZKs4YP0O4UImIr88ZEUbT4YpwgGa2A62dsF/V+HcJwe7ZsWKtuU0YlnlaRIjUBUYZoiVdqOd6p8/MdEPkxnBN6imzTEBp3B4c8CvD7NdzD/G2KjH2VRn1FZxHBekJMKqQOpzYYmyRxX+UGgKQFtAsc1K0YvkJ+8FOnTovhIYQHhNXoWJ8kOqlzJOvwqb6VNZDDOZi+iaTtddmEwwc4UyIkAtoh+dbhrXIXbbqs5czzPdYSzB3rtq8G+MDdU2yLqyjL+SkudZjhKTBEHR3UK5l9dSTDh+lXYzwlxK6b76NajKTiIrlm/TpGbkOkSOR/Pu4zTs2lCsprh5wKJPCVaqMp1VV1P/uP1afaj+BzCpYVAw72wJbUCNXaRMpxDLDk0AqcBLiL+Vhu6/CNRKtEaKMfTQaLoI4v2yCKcnDMpOs8JFNcadY8+i+KhzpBmlJfTYMlRzsTbfA2yS4vBIdS0WFwFaMG1x7M33KKzq64kqF4pm7Da6eijGxXw1uOuW75TOi1dIaylaHcmsRhI4ISx0VeZlPp8ejXQOmS0b5fs44I4aNsq0yc3MEwHjVYNkZgPjh4ZbANAa4XdUw6t0I5Y36ETYX7vVXvB7vqTYBTkRXqjt9QYRkuyUmk26uIBn9P7IKMEdlEp0OqnQ8ryu1/4E+GeaHbBiZ+2xF0fwY6sm9Dv4rhdgPWFF+3rMGeAHX/2B5WRh+uAXz8AbBBqeOVasn/WMrNgyqM4TH5gZ7tl7ybK/3iKTjBPrkWcVYuXWyBzmvv7ItOcDDGPC/ZapL4IzPN+ZiEKJhBZtKM0EhDokPEDqGLFM5UNWzXdCWDJD/G3Fj8AB3DBuOmTXOKmPHpGVsvoZAxxpAwPgbsBwIWlxcJlUd4OWzf2PfIVoIA2XBiKgeNy+dp6f2Fb5aF8KHfefDmc2zE+Ffvw85eohdawaNq3AFq3r+VV4PtqJ5m1KXIbhJ0FvQRTTWsB3lUA9ZmL1qgxWgqe4x4FO4TECoACGVbSlzFNb/1RZIbNBEjqOHVKNpWCH4Vm/4bbGDHf5Clx8K4B1/8R5IPhd8bggbi2yfBYgohCYKpdJTd3llH61JomOyyzMHwhnHF8qD7a3PrgXeaFyPeBJWL1AeG3t0GYgVpKsqfGMNOG+KmMEDmUV4aw2zKyRIZ9UyGJtGdCNNfz+1dQOYNVr4xTTzFgr5IsvUnDe4XeOWcEHx/5ywNNhsX7ghI/o2TWVn8xQGtmRXQDk0fl/8+WbCHy2xtvXz8ewnc4aOlTkr58e9iu0ZWi+8G0uYNRc5zltOljwN4E4Wt2zbhN0EMwW+VIGQzh3SBhDbwjGnYqfBPNBjL1H1+Xi7lzTz3QxQRiUZ8Lg8rCaw9bNj7XyF9S7YiOuOhGsAmjlJM48EZ0AjM3Dy0jXhhkWoVzvVPgTLnm7QhoxxxGzpxOPPajz7a4fgMG3+vhLHpE5Hiru1hYjVAaKqh3sSW1VuCxvqc0QuVU84SGcWqdf2TT0JiIOHhqe0uOpvKIjKN/5EQdGAJ6DGnMMq8yZ1KPRiMUhRO9rmsmrOh5+8JMSm4T4TWJpzDx/S0GgkFvCIBdOZBfsdcYSw3TBBIfDRsPrPMwazm8+emC7Y9ZkCwqo6iizNfFc/hdMADGt0l23f6AENFiCC2MY/b6jpSLrKbW6X3e2gYoPdGYmf++Y01R7rxkaWdGzjHPhKMfAdF6NpwkZvm3ULZsfbtHaQZ4lRnQj6ltazS3ALMQrglrTMUfYUgp3sxWA/sUr1PDUXHcBA4SmLkjgS7yvtpfPsQbk+/SaoNDx40aQsQWWpTv4tGLaTsDn+NYf5dGt+Z6hzAhPJtXAXaD6GPpi6Y8CWc7Cok3AYy69fJOyZ9JwanpTBwbQVsxN2ikQL9BmhE5SzIIWb8z9ahiJThc8GRNB2u+TlCrMnRmxNYKIwjeDw7q0AaIa9gvgKbL/0LZgXU6Tc7Wpb/CsHjXC2FPJwtHBYSwGRUbJR1QRw+wSndUE8jV0Yo2iI9Q4oF1NBZDMeRUE/NCuqA22ivSMWTfIYAGCubJSX6uXgiNmu6vMSgZsRnpNcyFN8jIF8XMxjvD5jK9X6F3CUvo1mtg0DF17EmPTl+qbcD+qsTHDzgBM9VASbvH8mdc0YRNoS/e6P4IEwoCH/bT8vwY6RlMsfBzEiaq0Dt8KGg7Xgv0oZCu2ZTN8U5/Yfb4LqKRLydXt5RKLSHSJ4zNlvO2bdULcfmsjAF07PEHDNhh/FRW/gbSBjDoLd9A1QO4S+NrqyZOyPu6UBbAkLOnbLMDvDo36cC3QSmvEgUUsIEgF/KI+1uaTUZOKDpqs8YaIgX7Le0LQWObRzeLqejZFpPSmOQ9hDqA9uqL52TwF6PSk1ZFchPCACEBBW/gPVsIzWlg99PYGYpICsW1KQYlLBeaUtD4N6NSRelb2PJNPmrCjjzYlaLMwR/x/jm0jQPOZhUiQdzF1sD6anJJR4lCRkA7PnKHk9k1yEVsMzh7NZJ4BsDrDEPlaaeSPvh9cv3Nr+bCmBySbgOWDWHXqXrdzqA1YEBf7rohfEhQvJ/mSNp1tgiDBWTz3bRCqPcaYdq6yHTIhFy0+eRoaCmp+S4K3MKbaIJ0DNgw0odaJXGGYgd+/7efYtvOYeEBUvLWq3NEufLpRD0hx/MHU1ELeF7GIb93M0MgikDJBFSzIfJERVyR7/cS89dPyHIpfQc6aRA6Ja+pzmWgp4ZgHd6FyuXBs+BmuKbmD/aO04w/NXM/iUtU2GNdTRNm6wI/g7YCqqlojo8ksBmA+smItSCW1FtiH1DKMvobI6AKe+uOsoZZhLVUJ6yxsq6ORSyuQvzDRu4GfdLP3kSpd3DNFaLUyswqnFMaMJePrV4Rm4MdhXD8lAe8SAi2TQtVtPXeCgT6nANPPCCqSBAymZQd6pBqIDZxXT8rWu1qSGD0fiZTGozqQfDUCoAobYd5+dlcvspF6nEhj+wQ5pPGpGHN8UoTcRmPlFOYkqQLuFUgZ0BHQC8gNv3iLRqfc8/aegqQMN0ChqQu6D00BG3Gf2+2nBkxcAIxYu3ts8pa/iRcLj48KqHeQdxbKDQDrjZJWbH1dfaaNu1v4K+f5AxGa3b7mAY0M3HbxVXYP7rE9q8Yhes7X58LFcFLsTy+ixqrbM+xgGAcY1KK51g4b0d0vQLiDuRu/7WCzjmUM5NlRSubKxpfB7Dxs1B8uwMEMmzsjP1KVQIQiQ0ftmntLMwhSMrRNIQm9R8kHvN/vCNc5Xzwi39D2gJOj1P+YMf+I9bCfoNuJxK4DSfdvvSDLhNiOeRa7q+p4KAUNgGzHOUD/OHEOVd/I8VFCxNtNUHh6U7+W6KPBnDetY6MLj1W8nKyFhOIsHUKaRNL3vbKjuokR5BSuOoUjjlvtiW4C8j1zLzP63OTeAvBZ7fT5JJpd4D5z43Plru89ez9Za+dnjVF/vXuNrsydvbEW9tWzDM97Fd9/ik1jDqfWJvbV5UMb8nEYfTc4I5BSm2KptMHbuSXNj3tA9WM+oafSMwZuhvcXv8vpYtGsQWFkaO1kD9NTv4CWJ44iKn6g02OUsB5/c6aMj2qlU98uPEVvA2irtqIK8Zdv9mg4SYU06Iu0HWaWNGP6I1d5tTUcPcJ92zUn1u67rRjoRcyJ8xe0ZFmF5zl2PRiA/XbFiT+wUExPhmhrUpafYnzb9X/i0F0KpNz2y0UJx7+W9Kpn8fd6p6SD2TtB0hRBeVeQi7M7y5Sdim2m+9lckSVntEOrYGzuNP8HA0LKyCnYSNsrD1xevWrxLWsM99dXSabWEiQaUrmebHUm1DNPV6JHpK3W/LAAb5AxxhcDT1TNjBVnr6SAtKTjRBGT0g+y9Z0Km5wMHLG6kE5LRTv7d7tG7edHyCpS5ajcgkKUUYTB2GXgqcijsfJMUAUOKSfLq9vQy4uCvrWt30HMh8e8aPBDoGPsB3XJDOlQAYCRLgf8AjE3wJ+qDmzAgtgfZ0Mzo2jMPJZMwAPmwlzzG8MO1CXpHAk8T9n8JvkLfnIf7Q0UsbVv8MtV2urOnxy9c98zxpxfgGvcNTd67C4zAfQlD6KEKaFwMkFuymr0b/OM1vzOz8TfTPs8ySBSQdy+y3bKOe/mrj4ddUHGUk1QiSZPvmUvAGeQaTkCQ72XH/B+GM+OD+FdpOHGTk8G0B0Kyr7ZCQDUQ9aSnKvvRKDVqWq25d2wlvDgcdlB9N7NOkpQKk8fhXp1bXA5Bu6hYFBl3ezp6Z/j4+Z/SVyqOlV4pMXCbCYvfXb9ff9gnsbVC/ToUxlrkS/U4DjPDZ4Z62gzf+i8fk51h3VhF/FEKzlL0TIAx8W5Ei3pNHPWAu0vfBkm5TN+qV4npk2EVUHZk9KK2KoyiIv+nHX33yXeVJ/UHd8L5LaLlM+Wp8uvYMG7oqsW18u6sFXzuHu7uapk02iKUS2x0tsHNqpRKXHxdTwH2vEo2AajYVNrfAC/xe2+ByZoRH1DtcubuzeDmqOSKrBaDa26apKQxrhK9mrJDwJ4crDmt4XZVnuuMR1D7yQnhmkjwBojp2Tug+yZwv2nW+EXFTGXCy0hUHGsIRZIBOwMFFOapzPYmmgDs7TPGjLx+jo0gFJNU6NV5wlL/G1K6L6N20MR9UgGhejpk6NvqlNGVFzwL+VTNpzBdR9XR8GEG0D4NEPQfqJSqxMinwfxBEZIyhNqZ2Yc/FFDwaPwf57R3vcRxO7HUqkbj3tNNxtyqNs3ToGBjGBt0FkBFd3Kx5hk9iVKFnVsNkljRqa0vO4dFkyHD9VDW0qTSShmvD1P2NtlQhLFtsVhjEnB+xxptbEcDmtwtebBDs5axw+9NlI1xy+GN2FxQjDG4lcGpUPxP0HpnxdFkFBuS+uTjC/6s5CmxO2jZeoQbfc7bbsrHYVFc/8r6CPUgh8VfzrHXD8TR0cXnr7otuTYP4xxDzEsj8o4vJdW/ZSlGwNmHL7SQQZLBgiYaEizc70d0UxHTHKl3EDnzsjSKfaJ2v8RKnuYsK2xn6Hkjc6J9VLLt/c+59fT+Vnbg26GoTvVR5HsS2ppBOwNjiNK0pH6Xz1z/p5fBuDdQS4nFwK7LxWWZv/5hu3+UGnbKHj/W9eRxLRqnPZBWHOdzcvEWRzUkCNiADdnmJwSBuPmaa+ayLcMg5WnAD6NX+2t5y8qdJAjo7oCiVX8mDFWg+e1Rd9gtcYqn6kRfp6ohXgXvnIj4W+8QYKnSFa0vjA/cSVJrc+AUdLebFtvHFLUNo6T6A8tmo5r+NHEUMcZiyUhRmVyAuBy0A8skGU4dzOO3Es01F/+JF1Osd8BdsJAUvHeaV7QUVwRfzkaTjhNhE8eiVv7Yc+ivO4KbWKWlFvN+oY9xBlHp7eBtfXT5nJM40ANM/nMPvj0W17I8Ma8l+sPnl+6ZRpkaN5nyA/qPn+lYVuGRQ4jAhexn4TPSNWOYH/b9p9Dbth/FU8n3R4Mp67PU5/BeEuZkbIkoPkdRBzUyXnb2G2/HOZDJ/5n3sDYp8TdXwM0DgWUHnGG9JArGhhYZhnzYYZ5T04YuNVrDM5mJvIrLRL/3z/4/EXLhpIxPlE/L9QkbnyYl7WQVHbP9ytfGNXrdL7PKlRcDhuE4+8ZiSY8TZz2PD7U3sysTsKcZ3GD3eoDrBwFPnBIiPMjXnmd6Tgyu8ktWfPdQj+3dnT4gkjJqZ0XI3SNWoTbBjv1NRPb5U3KZOh2MMA0tnoegpG67ckUcFK0ruNmSWJa7IMD4r1wVzJZ2mFrmgtYjtSgsvqEeenGP8NvDzhBZPbfbAVaCdFbLb3XWB2NK/Ju1yzU0kPSUfLFqPiAgRLMz5zl6n6HpPPjZzlMjl5bGmyvc4ToVxG0DJdCnGPN/WhQXx6JLDN4/BBp5+rYMRmE9u4mTzv6ggTEXi6Y8z3oYXmWzgyQmbNvxGbP4RwJGQgXKoqQLQm4kRPo/QWqyj5h3m0RNySzAT7rc1Vh3MgpHEkkk76GSjH6RVA+mhoSRpSmXFKaRKOm/NG7VKqnLHDyn07g2u1B+N42GpRrDDKLjBklJBbl4qdm8zmwhUOKHB7X6RldtMXqKOmUQlGDV/AGhgHBA8aa6TREWBAKkN0Wd9yWLB/qDzfF4qvlMuUQvPeOwWbOYVDOOAw0Q0ZilgYyhsApdSEkg6fmTkzgBoBYI2wT+6kuwqlmVtwWPu10SAgBVsEmEkMUkUFTyPv1LnLfbsB4DrbyGw9Z4HnEktP4cmAyFLKgQMyoZvWzPC2lglV8P5tnIPgzXhVJTuzUhCCVN3LP0ji/QDHg3XjXALsa9MzJRvz7AbsSuW96svHnvR/uBUs3SZzvRrXNQOIM/X8X7R1P8/gkDe+PwBhxBxxoFmMXYUkOvZzVE1cn0lCm7L7fNk3ecIdd+KCeB2QXlsX48J1iL5vdYHDvMMmGcXHD2VE01kO+9S04dGqWbsV66JVodcMVMtLmMg3F6Y2/ZPlj4P2zxR8M4TvuTcuJtf/F1pd2rv/IU6a+Mbj0wYF3jLRzqHp4h1+d6kGqMY9/e/x2kYMqP9sizDoXsRyOj9m0AbQpTO3PZhTbOT5EE3jDZ79tTJyMmPmcCDuh6jvlFGxfwp7GPa7SZPIl3Fj/7U3JpwhvvUWYmrBd7cwzIz1aZJGGhnCwuHM7dl7p8x6R3T3Nc5ePZFPd/5WXvqrUG3OY1KIwzMt5w8NC7Ni4dcbSf9HNzm9BzHRGzu3yo9LbvcOqDSGZ7W4uLuwFt+PHPBKPoyW1ZVrMhZi4K0nW4qXkFJbT6HpOBvwoePP8L/fdp5xx/XcHD+bm6PdC/1Ds7Cz9DWLqTxYvoqP44jbU+FxlsWwYYsC9/ttu1Mz+ezt2Rms2UKqo8eBFoDHCzyD4eCADmPqQW69UIUmFOxaNPxRfcq2cBqpv+YkfPxsPvVj4Nl/h+9NFiZmR3Aj1ZhuSmrm2eWTC0pCMDGBaOSCFJHpWaNLI0U5oE64gZxfFFyW1iLSb3O7gnhbG2H5WxTGgqN2YVDTl1WmuuNVFMw6yx9o2HmLVrQ7tw+2umA7+NyCnk3uHRhqeeqvpc8drumPlKkMFPEF8hP3Sa/1Z/fyY/WD+8C1ekcepCi7A9vfYQ5eU8gz67XV5G6WX18cA7sh8CqPkLz7Phy4sjOt+81MGmzKdqX4SnkpNeQ1/hHlzB0nOLexT8g39+HD8i22cpqAezZEuBUqfybdVvHkGqQ9htlVFcI4TCYBwMQNa0fPjFaOBVD6iM/K/CzCpxr6UpD6F24w6lxS5NVb6b0nsIl5DR/e/PjPc74yTDq4tUvfb8b2Wsw8ApGrQn8ERY4MIjMI5yp+0eePjlPCvzUXJYm6f0f7lbGn89A4i81Yt4GSTHl8WTrxyN7CnSeT8lgYD+H2tgUh06xHZh8vUnTvhwQ5bkXo4Kfz/CJyH8ce/n/pb/KCW6ObbmCxj+nc2zGvvcEuvAp1cbZmNKkA+Q53Krhe4PT4lQ1fbbGyRUGRQm2NaKsm+3PUbj1kKaxmdnY79bvfAGs6wuz9ODKmBdaoOfCb6e5yyGiAA4C9u/TBDcVIbyaRWc2jxQRrz37hQ0uMVv3zfE/cdetu6rFo7l6v0GIxmLIm8V+SZYHMfPbqHh8MrlfRC5+LqTiqfsPwkjEnJ/7IZR3TzzDHoTpLHx2p0EkSH3Y6XeYXkPpjr9Tke+LHaHezKbvB/dMaFW7zbYvdiff0Jjv+JMPv6LCDqC+6mMLrxPNSXThC4zTOMr85b9pMv13JZL5l6NhtvjCTH5DphO4sLO8edGnpK1v+sUbV6rl0KWxWZydZLxpJY6ExSHYx5ZtjdWhbvl95pnuF9rDXWvg4qfpUmU1hhy2ov0hxJGUIEkSMMC07CB3qWj5K3K7W9+FjYRtzFT9dkQ2OMs6uD9DKWpcFLlawmzbMK2TTdDzZGyYdGgkQE2aGQRHpy71LxXm16KjvW3/mNsBpXCK2MKoWR5f2XT3W6ELXwC+n9Is5tv/QhUzz6EQuMsfbxgfJ3xVgQQwVj2k6+QC8t2pSNdAROg57Jq2Ud5M4n43F5p6EkDhnlwXrNrPk/MFoMCtBibmrjZwdpbuxOMGTp2rzqOXc54PoJOtPbZL+SwH3PsgSvpdvyzwXsLFNWZKqcCFvqvQk/ND2Z56KM6QNxeJCnEEq+eRGNJyfy45wrcB7WEL2x/Pfk/vWTFrnKPmZTuT2PoqkshaA0ZtZIqJWyOZFRcRmUQcvxpG37s9ZaZfVzCKEfCYJRZVREqBgS6BkFtTpIilAOKZZkkSZIkSZJSzlBKkiRJkiRJKfE/TaHBGCi12TkAAAAASUVORK5CYII=\n",
"text/plain": [
"<Figure size 1200x500 with 2 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],

g4IMPJj8/n/Lycs4//3wWLVrUrebp06czdepUnnrqKY455hgKCgr47Gc/O8D/S0mSPmjzNL2XXnqJ888/n/LyciZ
Pnrzd6XttbW1cffXVVFdXU1BQwAknnMD8+fOZOHEil1122Tbv3dDQwD//8z9TWVnJiBEjOO+881i1alXX837f0FD
iSClpF9x77738wz/8A+eccw4///nPyc7O5ic/+Qmnn346f/3rXznllFP49re/m6AmJn77299
2m9YxevRoAC6//HLuuecevvKVr/CDH/yATZs28Z3vfIdjjjmGV199laqqqq7XrF69mksuuYRrr72WG264gUjEa5C
SlC7nnXcen/zkJ/mnf/onmpqaePnll7c55zOf+QwPPPAA1157LSeffDJvvvkm557LvX19dt9z89//vN85CMf4b7
77mP58uV87Wtf45JLLuFvf/sbAA8++KDfNzRkGEpJfdTc3My//Mu/cNZZZ/Hggw92HT/zzDP50Ic+xDe/+U2ef/5
5Jk+ezP777w/AlClTOOqoo7rOzc/Pp6SkpNsxSdLgcdVVV5GVlcULL7zAyJEju45/6lOfAuBrX/saU+LSeffDJvvvkm5557LvX19dt9z89//vN85CMf4b7
77mP58uV87Wtf45JLLuFvf/sbAA8++KDfNzRkGEpJfdTc3My//Mu/cNZZZ/Hggw92HT/zzDP50Ic+xDe/+U2ef/5
5Jk+ezP777w/AlClTOOqoo7rOzc/Pp6SkpNsxSdLgcdVVV5GVlcULL7zAyJEju45/6lOfAuBrX/saNTU1LFiwgPH
jxwNwyimnkJ+fzzXXXMPXvva1rj4AoLi4mD//+c9EoA1mD/+c9Eo1EAxowZwxFHHMEjjzzCJz/5SSZPntwVLH2wb3juuef46U9
/yg9/+EOuuuqqruPHH388e++9NzfffDM/+MEPuo5v2rSJ3/zmN5x88sn9/L+KJKmvLr300m7rBX4wlHrzzTf51a9
+xde//nVuvPFGAE477TSqqq46KKLtvueH/7wh/nRj37U9XjTpk1ce+21rFmzhurqag455BC/b2jI8NKZ1Edz585
l06ZNXHrppXR0dHT9i8fjfPjDH+bFF1+kqakp3WVKknZRc3Mzs2fP5oILlugWSG3tz3/+MyeddBJjxozp1hecccY
ZAMyePbvb+R/5yEe6AimAAw88EICl5f2WM+f//xngiDgkksu6fZZ1dXVHHTQQTz55JPdzi8vLzeQkqRB4uMf//h
On9/cX1xwwQXdjp9//vlkZW1/DMnZZ5/d7XFf+hRpsHGklNRHa9euBcKOYkc2bdpEYWFhqkqSJPWjmpoaOjs7d7o
Q7dq1a/nTn/5Ednb2dp/fsGFDt8cjRozo9njzNIqWlpYe61m7di2JRKLbFL2t7bnnnt0eb57yJ0lKv55+J2/cuBF
gm9/xWVlZ2/Qdm+1OnyINNoZSUh9VVlYCcNttt+1wOOyOvjhIkga/iooKotEoK1as2OE5lZWVHHjggXzve9/b7vN
jxozpt3oqKysJgoCnn356u2uCfPDYBxfQlSSlT0+/kzcHTGvXrmWPPfboOt7R0dEVWEnDmaGU1EfHHnssZWVlvPn
mm3z5y1/epffIzc31SoYkDVL5+fmceOKJ/OY3v+F73/te48WIrZ111lk8/PDDTJ48mfLy8mfZcyYMQRBwB/+8Id0lyRJGmTsKyRJPRmsfYWhlCQ
NYk1NTRx00EHcfvvt6S5FkjRI2VdIknoyWPuKrHQXIEnasTPOOIMzzjgj3WVIkgYx+wpJUk8Ga19hKCUpIyUSCVp
inQDE43Fa2jvJau8gEhn4AaT52dGM2LFDkoY6+wpJUk/sK3aPoZSkjNQS62T/f/1rWj77ze+cTkGOv34labCzr5A
k9cS+Yve4ppQkSZIkSZJSbmhHapK0i/Kzo7z5ndOBcJhtQ30DxSXFKRtmK0ka/OwrJEk9sa/YPYZSkjJSEARdQ13

j8TgdOVEKcrJS0nlIkoYG+wpJUk/sK3aPoZQkDWKNjY289957XY8XL17MK6+8QkVFBePHj09jZZKkwcK+QpLUk8H
aVxhKSdIgNm/ePE466aSux1dddRUAl156Kffcc0+aqpIkDSb2FZKkngzWvsJQSpIGsenTp5NIJNJdhiRpELOvkCT
1ZLD2FU5ylCRJkiRJUsoZSkmSJEmSJCnlDKUkSZIkSZKUcoZSkiRJkiRJSjlDKUmSJEmSJKWcoZQkSZIkSZJSzlB
KkiRJkiRJKWcoJUmSJEmSpJQzlJIkSZIkSVLKGUpJkiRJkiQp5QylJGkQu/HGGzn88MMpLi5m1KhRfOxjH+Odd95
Jd1mSpEHEvkKS1JPB2lcYSknSIDZ79my+9KUv8dxzzzFr1iw6YYHqwVFs3jkUmh/XnqHufVQpZHSDKz1t0NK3qv+Tn8NtvqXbRk
0dV5sQnHBl5KHuyfcjRi6KgD3vcBDyUm9a5Wva5OMN7GZeRzk/zOFA1i1bAe/s/V8XNYVHuGqkngEvXa3wY64YUyHuYXCIJhLxIBWoaNcNKpBT0UVIXgn/5LgEXATR0cUQPOtVeZ7oaEhsjl
sT4SfoCfLnvrnYmeDmEt9GTWhVwTlVlSI5V4+kJzoyCtk6j+6xmRj3WJJ5+YXkDz4kE7a9y9vzvlvIBqJ7aWE2/fRJt0GEWgtq2ujsaHzmrFZmHqp8wFajdbCRVrUmWR
mZdUNh55nupuzlqGXEbq2yY+y9E99ZdJz2SAg6P1Hcsw6Yn9Ta+dE7ZTVVWUutxzO7gLLCLDXXTQWKS8QY1SnYmz8xH9Lm0aEdPp6/9w+J9
5Kq9y5Aedi7Zt5lUVOIG2DzR3UkEpp7f1L3D68OSJC7zxC5RHbh3kz7NB6RwkDuTyPTHQFVT2P2N3POPP5+D9yH5uGFLp3H0yQnDl8Q6B8
GmXn3T9Re7fVqKhWp4Z6Q9a+4Q7kRmpKnh4SmGqHA7bKoRLqJXBNJIcJWyUpwx1PDDG1p7Hqd7OvZy+4DRtzzn6Z9ZWmjM6+PZyvXl
9j3tsuDNEy0z6dQtQ0zKE0LEGdkk9Kb4Oz0gZUXU7yVLqGx8DzOVHZoXpP2eG5TmqbEF4yMu0YR3pXH8QV0Lng5G/j1KHxAe6kNrKM
3YbO+lXl7FwU4K2gVDjmdbXJjPp3hYjd8X0C9iO3G4HovU7h0NmR8lhyEgUvC7F2NIs4hUMAqS6zQWxTpdwCPjWPp2UK4qH1B4Cm8LQJ
SPqPYqPH84SSgPpYMZuzpCxh4zsyr6QW3Yu6HDnNGK9fXMn3jRCMrEz9H0AOj5g0QvnP/Jb2I0BlcqrfQplAWxWOvQhh9mqbFeuTpmIp
PgRJzG3qgaFmAaKd+AB2RCPAuk8BJrjZahS6J5uulcZ5n0dQKtEHXUbU0ls/cU7eB9mrcdg8yiIC1cT6xjHJu/w7Lhd5PQZhF3HLqOd

FUolEgiuuuIIHH3yQJ598kkmTJnV7ftKkSVRXVzNr1iwOOeQQANrb25k9ezY/+MEPADj00EPJzs5m1qxZXHDBBQC
sXr2aN954g5tuumm7n5ubm0tubu42x7Ozs4fsf+ihXHtvZUIbwXYOJ8O9jalqm31F/xnKtfdWJrQRbOdwMtzbaF8
x9Azl2nsrE9oItnM4Ge5t7M+29SmU+tKXvsR9993HH//4R4qLi7vmapeWlpKfn08QBFx55ZXccMMNTJkyhSlTpnD
DDTdQUFDAxRdf3HXu5z73OaNa6++mpGjBhBRUUF11xzDdOmTevaNUOSNHTZV0iSemJfIUmCPoZSd955JwdvvdvvZ
uu+/msssuA+Daa6+lpaWFL37xi9TU1HDkkUfy2GOPUVxc3HX+LbfcQlZWFhdhdccAEtLS2ccsop3HPPPUSj0d1rjSQ
p7ewrJEk9sa+QJMEuTN/rSRAEzJw5k5kzZZ+7wnLy8PG677TZuu+22vny8JGkIsK+QJPXEvkKSBBBJdwGSJEmSJEn
KPIZSkiRJkiRJSjlDKUmSJEmSJKWcoZQkSkSZJSzlBKkiRJkiRJKWcoJUmSJEmSpJQzlJIkSZIkSVLKGUpkJkiR
JkiQp5QylJEmSJEmSlHKGUpIkSZIkSUo5QylJkiRJkiSlnKGUJEmSJEmSUs5QSpIkSZIkSSlnKCVJkiRJkqSUM5S
SJEmSJElSyhlKSZIkSZIkKeUMpSRJkiRJkpRyhlKSJEmSJElKOUMpSZIkSZIkpZyhlCRJkiRJklLOUEqSJEmSJEk
pZyglSZIkSZKklDOUkiRJkiRJUqSkmSJEmSJCnlDKUkSZIkSZIkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qSJEmSJ
SJCnlDKUkSZIkSZKUcoZSkiRJkiRJSjlDKUmSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qS
kSZJSzlBKkiRJkiRJSjlDKUmSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qSJEmSJEkpZyhl
JkiRJkiQp5QylJEmSJEmSlHKGUpIkSZIkSUo5QylJkiRJkiSlnKGUJEmSJEmSUs5QSpIkSZIkSSlnKCVJkiRJkqSUM5S
yhlKSJEmSJElSyhlKSZIkSZIkKeUMpSRJkiRJkpRyhlKSJEmSJElKOUMpSZIkSZIkpZyhlCRJkiRJklLOUEqSJEm
SJCnlDKUkSZIkSZKkiRJkiRJSjlDKUmSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJkSlHKGUpIkSZIkSUo5QylJEmSJ
kSZIkSVLKGUpJkiRJkiQp5QylJEmSJEmSlHJ9DqWeeuopPvrRCvu+wygiDo9u+oo47iqiDt+JC+UnwJVMM3Z6Tm5uLtXV1dt
bWxtXXHEFlZWVFBYWcvbZZ7NixAk9cS+QpLU51CqqamJgw46iNtvv32H53z4wx9m9m9erVXf8efvj
hbs9feeWVPPjgg9x///3MmmTOHxsZGzjrrLDo7O/veAknSzOSoGNfIUnqiX2FJCmrry8IwnXzOOuqOTV++HDpYDeohLPjRGq3y8Tuhw
9rq6ujrvuuof/vKXnrqqQDce+9+j5sscff5zTTz/9XyVkOqKykjPvx5VkjwYQkVN8/Hh7+vsYVG1rUnZGaqMTG3tSXgsSyoUHkPzKoodJJu
XvsC6deu6nps6/bz+6Wxh16gdqfQKxvmRZJk/tH+TGzH6uMzo+zEIdSAAHfBqtQhtnbYhyBwB9Xn61ZnSXXdIrpuEJG9J1vJUVj9vq3
Fi/n2t7/NySefZzXPz588nNzWXNmjXk5ORQXl7e7XVZv/YjVNsmqtdWjDs0c+/SZqhUOk0O6o0MPy4kbrdV2pnFk6JmoEg6sWqcb4bWaz3tuHbybvmNmXlt0gDWn8
q7otMaCPYzuEkE9oIg6t99hU7lwk/k5nQRrCdw0kmtBEGV/vsK3YuE34mOAqwAxeJp
dOpXDDjuMCRMm8Je//IXzbvY4/9thjFBQUz9MGSEsB2U6L1H8S0PE7cjNL0DpuKUd9a
exubm5nSX0MW+oneG+88kZEYbwXYOJ8O9jfYVQ89w/5mEzGgj2M7hZLi3sT/7in4PptT5o9OjRTJgwEgJgwgwfYULFwXJQXV1
Ne3s7NTU13a5faXnXzxqrFu3jmOLhxzJgxg5g5KSkoDuf69DwJJSko5ftQD+LxWLmMjezjezs7HHSXMyA
yoY1gO4eTTGjwMaNGjgxS3SXaMNvG9Ndwg7ZQ99HuXlY6pE5nUyfL7lZBzBcrJKxkfXQWhb4tEqkQTnx5LSdJ3YXJ3Hh1trYuS3YWYMtOqGqkLexhyH6jM7TdDUhs3bmT58BxPzSGuhm
MHj0agEMPRGHezTnKB8s7GmR4tVxQ3ajHPm8AIAgUfAaMR2E1aklj0KUKWhOzmOxnPomBjsFQNlrQqGbFDj3eOYzQZC2X7Bke0tGVrmIMDo5ZXpnwTt7RSO0BF
bZl+xfUO59t7KhDaC7RxOhnsbB3Pb7Cu2byjX3LuZ0EawawncPJcG9jf7atz6FUY2Mj7733XrrrF0LSXCw3JRrrr3LCOzrjCG88J7hmePSrpUKhmHgz6h5d8
FRQUzZ87k4x//OJKKKNPjPpefPmcdJJ33U93jzU9gDJLL+XO+/9jQMeIelLL0Pe5ze3HU13SczBoJvXaa0JaWmaHDSvGdFuaz5UVTrhSmSpRHkTnnvkK
S1BP7CklSn0OpefPmcdJJ3J3U93jz89dJLL+XO+/9k9ddf5xe/+AW1kKS1BP7CklSn0OP6dOnk0qpkkkdavzZ8V//61x8xGvlvL13SEkfYuMM/zXMbUZ8Jyy+V1ijmBx4kBlKmdryatuZbAR8tl1bsfSbbezSv
xwQUX0NLSwimnnnnnMI999xDNBrtb2hnfZJltLNvkKS1BP7CklSn0O+J1dkK+vAbbtRUjTyWrjjGU00l4nvS1L7f3HNJkdTJ7FgdVqxt8fb/PBBXJ07PGf5RXF7qmi5CIMtwLx5WCFHWehN19QbI+cIyVGIPNKhGsGEqcYvZ9MwB
vSRoC7CskST2xr5AkRdJdgCRJkiRJSjlDKUmSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qSJEmSJEkpZyhlCRJkiRJklLOUEqSJEmSJEkpZ0qS
JkiRJkiQp5QylJEmSJEmSlHKGUpIkSZIkSUo5QylJkiRJkiSlnKGUJEmSJEmSUs5QSpIkSZIkSSlnKCVJkiRJkqSUM5S
IZaW7AEmSJEnSMNKwFv7wT+H9I/8JZnwPon0AbCVfC+Kb0g5oRRcVxo9H9uPaSrPsmi3LpKbVWl7+fiMsxmN8ZnTGY39TqK/cL4dYhtnHGKPh1cKvwKtQoJmLc7Kd7mjnS68G7IrJy9fM9s6bO07P5nEhXHuBRFKw5XEgT1I7dCZ9JZhX6AJGkYQyQQt1JUpeUqfxmqr6tUHVz2NPRNMnuT4pSkLzeX+/AgZtPhTTM4kLqhjoZKIT1ZXjFzsk2eFTv7G1UPnHFHGLXEyLpzFMGhXBBZVdzebDpUN
p+UoIJakTiEyRsNVfYLm0UNKeUmsJ5UrtZ/PxMI1mvAPZsDpGzJyYDqQbYDWTZCpvqxaJ9tUuPd56swH7bGeDhKpYXotDjHUUypwJ0HUqy8IpXN2eFNhkM5ZTkwnjOxj5NLjXvmGd5zwq/+MO2Jj9NKh7Cvo/cqd5e8h6XTctMlZCpTDGMG4hgUQ8aoGJKzYK6YvmJKjYqhRg8k1t7g/+CiWT8R5QGvF5s+6nqmO9dVmAMmZBtmG8F0ttLinoTwx6MZUEpYm0Bo2f/8E1YCZ1qK8ktwqO5+ChbkLSx9s2zNGkIMSE6SZCdG0vx8hdzKFx/PEpADZe+VjVxdKxGyxmhvNZkEjvdj4NmSnn/MSKM7pvwdnkP4l6Yrm+LIhQhwDhtYw0mEcR6CwZnmNSP5vRV7iCyVnfaSNMFU5rjB3s9TkS0PtbTgC3nh0nH7zEVx8P3O6KNFZtjhASnpBtl+UZjgnXpEO9HkDQn3OQSC7LdJKWHcgmnuJcDsFqHYnJpqm5E+8Z0hNNA+/cxOPA5aI5bLVBT00wOuWB0pPr7bOXZ5SnOxDi+N6Pq0JmvRLkX8xQkq1vT44oxEN1G2N6AFVa6uOJbvlXgeVEeh2b0nRF5a4I+yRy4fpH0h9BtpWCbSm+6Ipxe5A8PFX5eXrXOAiOgs2WzRnH1pcW5FhAfeFLWeDN41Ytk1iF8UT8S20h01lqcd9rlc4NPaRtiMjhKqGbXF4jnTKoRG7lkf0YqZhZm0Ui4qXFtOjJc8iFK1BiAGmc3F8eXTeeS1ic0PCVAv

kqdfiyel779ZFmf3O+u5PjtoPckuhvRHWvJaG6iRJ6ZYXDxctzysq2+a5ySMLqSzKpa0jzqvL67aavmcoJWUqQyl
JktQ7iQTRzdP3Evn8/uUV3Z+PRGH8keF9p/BJUuaJxymgBYD87YRSQRBwZHK01POLNm6ZvudIKSljGUpJkqTeiTU
TJDoBqKeQx99cR11zrPs5XYudz01xcZKktGtv7LpbVFK+3VOO3DMZSi3etGWkVO0yiHcOeHmSBh9DKUmS1DvJqXs
diQjN5NLeGefPr6/qfk7XYufPQSKR4gIlSekUbw1H07YnohQXFW73nCMnjQBg/tIaYoXVEM2BeAzqV6asTkmDh6G
UJEnqndbNO+8VkBUJ/4T4/Usf+BIx5hCI5kLzBtj4XqorlCSlUXNjLQCN5FOSn7Pdc6aMKqK8IJuWWCevrWqEsvH
hE07hkzKSoZQkSeqd5EiphkQ+Z0wbTSQIr3Qv2dC05ZysXBh7WHjfKXySlFGa62vCW/LJy45u95xIJOjahe/5xRt
d7FzKcIZSkiSpd5KLnNdTyD5VRRw3ZSQAD778gdFSm9eVcrFzScoom0dKNQcFOz1v8xS+FxZvcrFzKcMZSkmSpN7
pGilVQFlBDh//0B4A/P7lFSS2Xj/Kxc4lKSO1NYb9RFt0++tJbbZ5pNS8JTV0lk0MD9YsGcDKJA1WhlKSJKl3Nod
S5FNekMOM/aspzImyfFML85bWbDlv3BEQRKB2KdSv2sGbSZKGm/bmsJ9o7yGU2m90CcV5WTS2dbA8MSo86PQ9KSM
ZSkmSpN7ZavpeWUE2+TlRzpw2GoDfv7Riy3l5JVA1NbzvaClJyhgdyVCqI2vnoVQ0EnDExORoqbqy8OCmJe7aKmU
gQylJktQ7Wy10XlaQDcC5ySl8f35tNa2xzi3nTjgmvF32XEpLlCSlT2dyl9bO7KIezz1yzzCUemJtfnigrQ5aanb
yCknDkaGUJEnqlUTrlpFS5QXhVt9HTRrBmNI8Glo7ePyttVtOdrFzSco4idYGAOI5xT2eu3mx82eWNpEoDkfdOoV
Pyjx9DqWeeuopPvrRjzJmzBiCIOAPf/hDt+cTiJgqzZ85kzJgx5OfnM336dBYsWNDtnLa2NqNq644goqKyspLCzk7LP
PZsWKFUiShgf7iuGps7kWgPqtRkpFIkHXaKnfv7TVLnybR0qtXQAttSmsUtJQYV8x/ATtjeGd3J5HSh0wpoSi3Cz
qWztoLhwXHnQHPinj9DmUampq4qCDDuL222/f7vM33XQTN998M7fffjsvvvgi1dXVnHbaaTQ0NHSdSdc+WVV/Lggw9
y//33M2fOHBobGznrrLPo7Ozc7ntKkoYW+4rhKZYMpZojReRnR7uOn3vIgABmv7ue9XwIAvAxGUjA8udTXKm
kocC+YviJtIf/baJ5PY+UyopGOHRCOQCrgqrw4Pp3Bqw2SYNTn0OpM844g+uvv55zzz13m+cSiQS33norV199Nee
eddx9h9azzTk0GRr9XE/+c9Gv3Iz37lqscBDgPAdyNd8U7L+PwRwPVstsLU4e9E/WP2ZgFM0dtpWM6zHmbZzB14SzL97/IFRnZB5x7sKsXyitsFJnD0EPvgRq1kAdUmj4NLK7NFqkWTGAxODA5BWJkHaoRqXKXfuhovqTrvrl1OBpSe4M7BSPiJsRZ13bb4
SDsY2DgTbOXxkQhuhf9vXr6HUmjVrAKiqqup2vKqiqVLl3adk5OTQ3l5+TbnzHyQlyKEVqVnQXiNbRGH6ehkR6PCJEYcOfYT9MxH0PfCFZj9rbrqZG9EdNpeRAskllsKd7Ggk+74Hel5v6ipLL9TWyTkdgOmHf1zGjvB
o6I/SU27WrFnpLqFHQbyDkQ0LGFP7IiMa32bRqNNZPPK0Xr9+KLSxP9jO4WO4t7G5uTndJQD2FX0x2H4mpzduBKC
xHR5++OFuz+V1QHYkyjtrG7n9gUeYXAKFbW2cCrByPo/++Q/EIznbvOdga+NAsZ3Dx3Bvo33F0DOYfib3i4XT9xa
vWMOqD/QT29MRh5xIlFUtWbxddRj71T3Fqr/cxMsTvtDtvMHUxoFkO4eP4d7G/uwr+jWU2mzrIf0QDr/94LEP2tk
51113HVdddVXX4/r6esaNG0dDRDkjXGMsGzKVrswKxh7JEaMD/KJRfbLUsLEn2YD1zsJzUH0Oh6jOmHjnLsNSnZ
/YJTKBaLMWvWLE477TSys7N3XuIm1wKALcgxV7gZ2utCdsvdEn/rT3hCjXXPbuBHEQz3+yCtgw/NA3P4MIz7bA+VjHfFkHtPCsNLR3XD/80EM+rTw0vUHJw/SUWD3NkL3lu7Dm3YcTp3XQEB/YXTDB/RKzbvOkxJGtDe/wBh3b3H0NUYZgqAfbzvFT4bEztPFH6ErBRfWS2ysTzvkiGmFfSEduk8wRI02mx2Pn1a9xD0WSdZ6i1NB7qWuOc9+U/ecH7YpZKzAR7xWNhsujTJC1R4Wmg4/AUCdHWaHbPgj/5x6OC6VvKjQrtBKMkWMZa+h8z/Pp/3Qt2xJCB5xEzEZKlH7WzNqn9uqLudTVfN4ZQhTjTeNAEsuhI5uN7xH4OIq2qdvhZd1i5X0OckKE9Ja4OL6uz6FhAhKSZ72zwkfgTuYH1N3K6kB16Ln9QOPTMF+f/1HVRmLBpwegt14j6dZ1cDuQHnEgbq8ZtXTZRlHc11dhbg5qyaKQSrOP1Zq4nUmVWUlHoh1kx6MYHfT3+nS2pNeE/QUAfEpMc2o0fSpLdz2pL9gVVdHcw2+e4rY8qQeqKETSPVxEVdGf7kqzTXY9XcHcMUhWdMnDHdKMgbhKDe6/VMMWZTRlaWpgR
oz5tG3DMUB5zGNRdLnJXFLx8Ejme8PHq+lGrWR9THhYHkJmwcTaI0MFnVn3WjJVbE2PKfgpjFNP0eT2YP/LPiPNeu0dzrWIhPZK4pEFB5EJSvYuu/vZy8IUGk4HvLxeVHv+NUhDMjL3CjP8h1/HTpKp1TOr4C7vq0nHLfVgvqN7d92lRpGq5OROVnQ2tr0QLf5MwRFVzQEgIpy92T4dwDtGcgTz2vuKvcf5JvbBRcRHi2gCzmsHvmgpa9dMu4l0MaXHBY9yHqJ5YRUlLKH5ly0QHlrmO6qMexNDfE+MEWsECe/pFl6VdA0lnHPk4v7QZrtjsMZyeAE8UYmr12VJ54UsZyYhyT15p1MK1N3RJI1YCdPnsBL0YXdvsUS+E5TB3Vj1NE6L0JoGBqaimb6EsTdSjVXXLtMt22BbB4mTsOoVm2dE7ybM8MhnN6MtzJwxyvSmMHv4qI11VjIGZKm7IaYTmBcyx6q1vAeDD5WbNbfP6Myl8Qw6SZDUcHyl+TyGnQVTcqvbZTdb0s3tW+3cX2TDaNCTZrNx8bRxcZOdkP8p+Oo59HHB6JxHbvFvS6HqP9yNHcOB5T6ADbLW0eP3BLJMbevD5rI5sSA/IzJcc2ya52F0PEE2X8MH2b8iEFVpT5fPcsEj8kJ/L7aaJr+F0ItBoFcN5qUQDh9cnBD3bNCsGTBpqhAE7YkcNzXPCLQ60vX1Jw4Y9H3nkY5ixTqKEp3PbFxrP/pRTaNQDJdsJO6hQGamnD0AqOFb+v1qs3QPNbtwHx5Z0JqyCn1HLxjP40KMtB0oORIbQf0DKBQ0zv/sGPrHHU4OD/p6kZvyfpGQaJ9kiBmOzYXhLrG1+LgZ7J4pIXSGe8RK4BUUJ/NGcwdjpXjBq5kK4Z3H/vNl3+Mb3b7kKaQf8lSUHjZgXm0JDHeSlB6FCugdHRbJdj8gXZPlzDT5D+sG0L6TDi1EaYlA70lLl4K6cP0aQdiyKuQsMTSDjCGx28yG+pE5qq6wahVXw4dUxpUJakQc84Kn3G
ocV0pSZnNvmIYaq0DoIECygp2PGT7IweO5t//8iYra1v4+9vrOHX/KtjjMFg5D959BA77bKoqljTI2VcML40NtYx
I3o/m9X59rrzsKIeMK+P5xZuYuybKxL0/DG//GV7+JZw8cyBKlTSI9DmUmj59OolEYofPB0HAzJkzmTlz5g7Pycv
L47bbbbuO2227r68drIDUnR0rlbyeUGns4ZOVB41rY8C6M3Ce1tUkaUuwrhqHWcKRUfaKAsoJtd9HbLC87yoWHjeM
nTy3il88tDUOpfc8MQ6l3DKUkbWFfMbw01dcC0EIu+ZG+hYJH7TkiDKXe38DFh14ShlKv3g8nfnMAKpU0mPTrmlI
a4jZP38vfznDbrFwYd2R433WlJCnzbDVSqnwnoRTAxUeOB2D2u+tZurEJ9knu1LdoNrQ1DmiZkqT0aGkMZT120BDu
e4r0jx02pBGDu+xuJTz4ViqqheQPBwsf6tcb+1hlP8Le317KpqT3dpUhDlqGUttg8Ump70/cgXFcKDKUkKRO1bT1
Sauc7rkwYUciJe48E4P+eXwYj94XyidDZBu//baArlSSlQVtTePGiNdL3UOrgcWUU5kTZ1NTOW+ua4eCLAIi8Mrg
XPL/h4bf47D3z+N5f3kp3KdKQZSilLTavKbW96XsAk5JbeS+ZA/F4amqSJA0K8ZberSm12aePmgDAr+ctp7UjDvt
8JHzinUcGrEZJUvpsDqXas4r6/NrsaIQjkrvwPfPeBkjuwhcs+ht57Zv6r8h+9NiCNdw1ZzEAb6+p7+FsSTtiKKU
tdjZ9D2DMIZBTFJ63bkHq6pIkpV1bclpGfaKAsvydT98DOGnfUexRk9tc4w/v7Ya9jkjfOLdR6GzYyBLlSSlQUd
zGMzEon0fKQVw7F7hFL5n3tsIlXvB+GMIEnHGbZrTbzX2l+WbmrnmN692PV5R05LGaqShzVBKW/Q0fS+aDeOPDu8
7hU+SMkp7Uy0ArdFCcrJ6/vMhGgm61pb65XNLw/4jryy8sLHihQGsVJKUDp3JEbWd2cU9nLl9m9eVemHxJto6uF
DnwZg/ManIDEIZmm0NUJnjPaOOF/+1cvUt3YwdY9wl8G6lhj1rbE0FygNTYZS2qJr+t4ORkoBTVF389MDXI0k
aNDqawj4i1ocvGxcePo7saMCry2t5bXUj7H16+MTbfxmIEiVJaRvawhc/o+fQ9gn6piKotyaIl18vKyWtj/HBI

5RRS1ryNI9wXx1a/CLfvD3Wfyg0fe4tXltZTkZfHflxxKRWE4enjFJkdLSbvCUEpdEsmRUp/99SL+e/b71DVvJ+3
fvNj50mecfiFJGWTzFfB4bkmvX1NZlMuZ00YDcN/zy7ZM4XvnYdjJNvCSpKEnSIZSQd6ujZQKgmCrKXwbIKeQ+IH
JBc+fubl/itwVjevgVxeHu9CueIHX5j4KwA8vOJix5QWMK88HYEVNc/pqlIYwQyl1SSTXlHq3IYvvP/I2R934BN/
+wxu8v36r7burD4S80nAXpjWv7uCdJEnDTmsYSpHT+1AK4KIjwil8f3p1FU3jpkM0BzYtgo0L+7lASVI6Be3hd4a
gDxcvPujYyVuFUkD86CvoDLKILJsbbraUah1tcP+noH5F16GLs57gc8dN4rT9qwAYWx6uoeW6UtKuMZRSKNZCpKM
VgCC/gn2ri2mJdfLL55Zyyg9n85m7X2DBqjqIRGHCceFr0j2MVpKUMkF7uIBtkF/Wp9cdOamCSZWFNLV38ud3GmB
iOOI28u6j/V2iJCmNsjrCUCqav2sjpQCOTa4r9eqKunCNppIxLBuRXD5k9k27XWNvvLeukXufW8qdf3+P13/yGVj
xAs2RIv4996sAfCT6Al8/YVTX+WOTI6WWO1JK2iWGUgol15OKJaLsUTWSR/7leO77wpGcul8VQQB/f2c959/5LLP
eXLvVulKGUpKUKbLaw2kZWQWlfXpdEARcePg4AO5/cTnse2Z4/N1H+rdASVJaZceawts+9hNb26Msn0mVhXTGEzy
/KJzFsbDqLBKRLFg8G5Y91y+17khrrJMLf/Is/+8Pb7Dh8ZuZtv4vdCYCLm/9MnfVHcZbTCKHGDlv3N/1mrEVjpS
SdoehlELJ9aRqKaSyOI8gCDhmciU/u/Qw/n71dE7YeyQtsU4u/+U8HqqfHL5m2XPQ0Z7GoiVJqZLdEYZS2YU72Qx
jB8770B5kRQJeXlbL++XhSKlg5TxyY3X9WqMkKX1yOsNQKqdw10MpgGP3GgFsmcLXklNJ4sBPhk8O8GipR99Yw8a
mds4qeINvZf8q/MhJX+W40y/ghnMPZMwp/xyeOP+errURx3atKWUoJe0KQymFkutJ1SaKqSzK7fbUxMpC/vfSw7j
oiHHEE/Avf2ulKascYs2wcn46qpUkpVIiQW7yy0Zucd9DqVHFeZyyXzjV4d63OmD0wQQkqKp/pT+rlCSlSSKRIC8
eTl/LKyzbrfc6LrnY+ZxkKAXQecyVEETh/SdgxcB9/7jv+WVMDlbyw+C/iBCHQy7h5Ev/lctPnMzFR46n9PCLILs
wXBdx6TMAWxY639RMwk08pD4zlFIoOVKqhiJGFudu83RWNMIN507ja6fvQ4IIf2/bG4DY+7NTWqYkKQ06WslOhDu
yFhRX7NJbfDK54PmDL68kNuXDAFTXvdQ/9UmS0qo1FqeQMJTKL9q9kVJH7TmCIAjXdlpbH655S/lEOCg5WuqpgRk
t9d66BhYsWcn/5NwSXogZdxR85GYIgi0n5ZXAtPPD+/PuBrYsdN7Q1kF9i7uTS31lKKVQck2p2kQRlUU52z0lCAK
+dNJe/NcnD+YFDgDgnWf/Qm3zIJ3CF4/DU/8BD38Nnv8feO8JqF0WHpck9V5ruMh5PBFQVLJrXzZOmDKS0aV51Db
HmBM5HICR9QvCUbeD0LqGVv7fH17n5WU16S5Fkga9upYYRUE4fS2vqGy33qusIIdpe4R9zbPJdaUAOP5qCCLw7qO
w6pXd+oztue+5Zfx79t1MDlZB8Ri48F7I2vZiPYd9Jrx96yFo2khedrRrpomLnUt9ZyilUNf0vaJtpu990DkH78G
5510MwJT2N/nar54fnENV338C/vZdeOF/4JGvwb3nwa3T4IbRcOdx8N7j6a5QkoaG1nDtp0byKSvM26W3iEYCPnF
YuOD5T98tJFEylqxEO8GCB/utzP7SGU9wxX0vc+9zy7jl8YXpLkeSBr361hhFJHfyzi3Z7fc7NjmFb+77G7ccHDE
Zpn0ivP/Uf+z2Z2ytNdZJ7KV7OS86h3gQhfP/F4pGbv/kMYfA6IOhsx1evQ9wXSlpdxhKKbTV9L2eQimAQw4+jFh
BFblBB43vP8s9c5cMcIG74KVfhLdjj4B9PgKVe0MkGzpaYe3rMPf29NYnSUNFWzhSqp4Cygu2P5q2Ny44bCxBAHM
XbWLjAZcCEH3qBxAbgD/iW+th7QJ451F44acw61/ht5+Fv1wNNUt3+tI7n3yP5xeH/eKi9Y39X5skDTN1ze0Ukfx
dnlu82+93XFcotYlu176PvwYI4O0/w5rXd/tzNpvz7ByuS9wVPpj+TZhw9M5fsHm0VHLB83FdO/A5UkrqK0MpAZD
YeqHz7awptY0gIHuv6QDMiMzjxkfe5q3V9QNYYR81bYB3ktuNn3ULXHQffPlF+NYa+MTPw+M1S9JWniQNJbGmcAp
bQ6KAsvzsXX6fseUFHD8lvPJ8T+w0mrMrCBpWwfM/6Zc6AejsgD/9C3x/HNx5DPzqQnj4Gnjmv+CN38GLP4PbD4O
/fqtr6vrW5i/d1G101MraFlpjnf1XnyQNQ00NdUSCZHqUW7Tb73fohHJysyKsbWhj7dbXLUbuDVPPC+/31058sRb
2fuorFARtLCs7gsjxV/X8mqnnQ04RbHwPljztSClpNxhKCYCOxmQoRSEjCnt5FfzACwG4KHs2eR31fOVXLw+eP9x
fewDisXB4bfXULcejWbDHoeH9uhUQHyT1StIg1tIQhjf1FFCyG6EUwCcPD6fw/frVjbw5+uPhwTk3d43Y3S2dMfj
d58Ir1wD5FVB9IOx7Fhz5zyw//FusH3lUOOXi2dvhvw4OR812tAHhmihf+dUrdMYTnHPwGIrzskgkYNkmr3xL0s6
0NNYCECcC2QW7/X552VEOmxju9vpuXdD9yRO+BgThmk5PfAd6WEZk7uO/59UbTmLhr78Njeu2eb7uwasZ37GE9Yl
S8i64CyK9+IqcW7RlKuH8e7YKpewvpL4ylBIAHY3hfO3W7FLysqO9e9Hkk6FqGnmJVi4veJKF6xr53l/eGsAqeym
RgJd+Gd4/5JJtny8ZE07ji8egflVqa5OkIailIewjWiKFRCNBD2fv3Kn7VTGiMId1DW38leNIjNo/XLNqzs27V2S
sFR74NLz5h/B3/IX3wtcXwz89DZ/8P16d+g1OmTuNw5dfwb+XXk9bxT7QWguPfQtuP4zEwsf55u9fZ2VtC+MrCvj
ux6YyqbIQgEXrm3avNkka5loaw7UHWyMF3Xer2w2b15XaJpQatR+cdn14/+kfwp+/ut0LzZsa2/jd7d/gyKc/y0H
tLzHlzR/RefP+8Pt/hBXzwpPe+B2lb/4f8UTAL0Z/i1Fjxve+wM1T+N58iEn54Qip5ZscKSX1laGUAEhsvkKd14e
tvoMAjv0KAF/IeYxc2vnlc0t54u1tr0Ck1MqXYP1bkJUXDq39oEgUSseG92t3vq6IJAnaklfA27J2f52QnKwIHz8
0/B387PoonSd9O3zi+f+B2uW79qbtzXD/RfDuI+Hv/ovuh/0+2vV0TVM7X/y/l2jvjAMBd62dwoFr/5XHp/w/EkX
VULuMzvsv4fnX3yYrEvCjiw6hOC+7K5RastFQSpJ2pr05DKXaooX99p6b15VaWB/Q0fmB3bOP/Rc461YggPl3h2s
GJke9Ajzx2hKe/+F5fHzDnUSDBM/mHsNL8b2IxmPhjIqfnQL/cxKJh8LvMj/uPIcPTT+3bwWOPgjGfAjiMfZZ/Sc
gHCk1KDeAkgYxQykBEG2rBSBSOKJvLzzgXCgZS07rBm7e920ArntwAXXt/VxgX7ycHCW1/zmQX7b9c8onhLc9LHY
rSYKO5trwth9CKYALk1P4FtQEvFt8FEw8Hjrb4O/f6/ubtTXA/30C3v8bZBfCxb+GKad2PR2PJ/jqr19hZW0LE0c
U8NhXT+CUfUfR1hnw+df352PR26gpn0ZWZwv/nPUQV8/Yh4PHlQF0hVKLHSklSTvVkQylYlm7v57UZgeMKaU0P4v
WzoBnF29nivdhn4FP3BOOjn3zD3DfhdTV1fC9ex+h+rdnc0ZiDp1EWHX09Rxx7V/4xf4/46Nt3+X38eOJR7Jh1Us
E7Y28EN+H3xRewgl772C3vZ05ONyRvGz10wA0tXdS2xzb9UZLGchQSpBIkN0ediQ5JX0YKQUQzYajvwjAGfW/Zer
oImqaY9z7XoR4PA1XCdqbw4VsYftT9zYrnxjeOlJKknrUmfyy0dkP23wDTB5ZxDGTK0gQ8Kn/ncd7B10bPvHq/bD
mjd6/UWs9/PJcWDoHckvg0w/Cnid2O+X2v7/Hk++sJzcrwh2fOpS9q4r52aWH8aOLDqGiMIdX18a4Yu1ZAHw663E
uP3jLZh9dodQGQylJ2pnOlnDDo87s/hspFY0EnDVtNADfe/gd2jq2sxbsAR+DT/06vCix6O+sufUkvrjw8xwQWUp
TVjkdl/yRMadfSTQa4T8/cRATph3LVe3/zNFtP2bRgVfxdN5JfLn9K5x/xMRdm55ePS2stWYRo5KbRS13XSmpTwy
lBG31RBIdABSU7MIVgg/9A+SVEtn0Hj89aj352RHerYvw0zlL+rfO3njroXDr8vKJMOG4HZ9X5kgpSeqtRGsYSiX
6KZQCuPkTBzKuMEFNc4xzHmxmw/gzgQQ8PnO756+pa+WNlXW8uryW+UtreGHxJlY+9B1Y8SKJ/HK49CEYf2S31zy

9cD23PP4uAN/92FT2HxPWHwQBZx80hsevOpFzDh7DnPhU5rM/OcSIzPlh1+v3rAyv+C8ylJKknYq3NYS3Of03Ugr
gq6fuRVF2gvfXN/GT2Yu2f9Lkk+n49B9oCIrZJ7GY8qCRpsoDKbxiDrl7ndB1WlY0wi0XHswZU6tZ21nEh+cfwad
rv8CGoJwLDhu3awVWTA5v61awZ3kW4A58Ul8ZSqlrx6PmRC5lJbvwhSO3GA77HACj3/gfvv2RfQG45Yn3mL902+2
2B9TmBc4PvmTnO2d0Td9bMuAlSdJQF2kLr4BH8vovlBpRmMOXD+jk6D0raGrv5JPvzyAeZMF7s2DxUwAkEgmeX7S
Rz//8RY668QnOum0O5/z4GT5+51w+/ZPZ5C/4FQBXNn+O//dCFm+srOt6/1W1LfzL/a+QSMBFR4zjE9v5wlFRmMN
/ffIQ/vKV45l0wQ3hwZd+0dU3TKwMd5Da0NhGQ6vTMSRph5KhFP148QKgND+b8yaG60nd/vf3WLS+cbvn/eidMs5
t/TavJPam4cDPUHj5rC1ryG4lOxrhvz55CKftX5VcZxBO3reK6tK8XSuwsDLZ5gQHFYbfqdyBT+obQylBSxgc1VJ
IZXFuDyfvwJGXQzQHlj/PJ0at5EMj4nTGE3zlVy9Tl6p51RvfD6dwEHTN796hsonhrdP3JKlH0Vj4ZSNaUN6v75s
XhZ9++kN8+IBq3uscxb0dJwGQmPWvPPzaKj52x1wu/J/nePytdQQBVJXkskdZPhNGFPDpsteoCBpZQyV/ajuYe59
bxlm3zeGs257m3ueW8qX7XmJTUztT9yjh3z56wE7rOGBMKRX7nxTuKhvvgNk3AVCcl01lUdgvLtnglwxJ2pGgLQy
LIrn9s/bg1j40IsHxe42gvSPOtx58Y5uFxJ9btJHb/7aQ9xJjWXruHyg+71bI3nHIlJMV4faLw2AqEsAXjp+068U
FAVTsCcC+OesBR0pJfWUoJWgJU/3aRHHXH999VlwNB30SgOhzP+bCPeOMr8hnZW0L1/7u1dTsQvHK/4W3e50CpXv
s/NzNI6UaVofbiEuSdig7GUplF5b2+3vnZkX48ac+xEVHjONHsfNoSuQSrHqZ39//M15dXktOVoSLjhjP41edyPP
fPJVnvnEys792Ev9v1HMAjDrxC/zy80fz0YPGkBON8MbKev7fH97g5WW1lORlceenDiUvO9q7Yk76f+Htq7+CDQs
B2DO5rtSiDdu/Oi9JgqyO5MWL/P4PpYIArj97P/KyIzy7aCO/nb+i67mapna++sArxBNw/qFjOefgHr4DJOVmRfm
fTx/KazNP58g9+7jR0weNCKfwTQrWALB8kxcxpL4wlBI0hyOlahJFVBbl7Pr7HH0FAMG7j1DZsZpbLziQ7GjAXxe
s5ZfPDfCIpHgnvHJfeP+QT/d8fsGIcEFEgLpd3IJckjJEbmcYyOQW9XEzjF6KRgJuOHcaF570IX7eeToAV+f8nit
OmswzXz+ZG8+bxuSRW61Tsu4tWDYXgiiRQ/+BY/eq5LaLDuG5b57Ct8/anymjisjNinDrJw9mXEVB7wsZeyjscyY
k4vDkjYCLnUtSb2R1hL8js/L7/+IFwLjyAq48dW8AvvfwW2xsbCORSPD1373G6rpWJlUWcv3ZOx8V+0FBEFCUm7X
7xSXXlaruWAk4UkrqK0MpkWjeCEANRbs+Ugpg5N6wz0cISDB53SNM26OUb5yxHwDf/fNbLFhV18Mb7Ib3nghHPRW
MCL9Q9CQItlpXyil8krQz+Z3hl42Ckv6dvre1IAj42un7stc51xGL5rMfi7l6wiJGbm9a+by7w9t9zoCSMV2HKwp
z+Nxxk3jsqyfwxvWnc/K+VX0v5KRvhrdv/A7WvMFEQylJ2ql4PEFuZzg6KGcARtRu9rnjJrFvdTG1zTG+95e3+L/
nl/HYm2vJjgbcdtEhFPZHwLQrkiOlSlvCC90ralpSM0tEGiYMpURbYxhK1SWKtv/Hf18c+xUAxm16BhrX8tljJ3L
KvqNo74xzxX0v09TWsbvlbiuRgPnJLygHXghZvRztVT4xvK1d0v81SdJwEY9TQPhlo7B4N6c49MKMw/cn++h/Ch8
8eWP4O35r7U3w6v3h/cM+u933CIKA7Ogu/olTPQ0OOLfr8x0pJUk719DWQRHh6KDcAQylsqMRbjxvGkEAv395Jd/
505sAfP3D+zJ1j4H73B4lR0rl1S8hCKAl1snGpvb01SMNMYZSoq1uAwBN0ZLer7uxI+OPIj72CKKJGJEX/4cgCPi
PTxxEdUkeizY08e0/vtEPFW8l3gkPXwPvPBw+7s3Uvc3KHCklST1qbyBCGAwVlQ3M9L1tHH1FOMV6zWvwziPdn3v
j99BWF15Y2POkgfn86ddBEIG3/8x+ifeAMJTyyrckbau+JdYVSmXn9+/uex90yPhyPn1U+Dd8e2ecE/ceyWeP3Y2
FyvtDcqRU0LCKCcUB4BQ+qS8MpUQsOVIqllvWL+8XT64tFZn/v9BaR0VhDj+66BAiAfz+pZU8+c66nt9k9avwkxP
hie+EV8W3W3gr/OZSePFnQABn/AdU7d/7Qrum7y3p/WskKcO0NoTrDrYlsikvHdgvG10KR8CR/xje/+Boqc0jYw+
9DCID9GfMyH3CkbfA2LfvJgigobXDK9+StB11LTGKguTi3rkD30987fR9mFRZyB5l+fznJw4iEgkG/DN3qqAC8so
AOLQ47DNX1LjYudRbhlIi3hzuvpfI65+1QhJTTqc+bw+CtgZ48S4AjphUwWeSVzH+7aEFtMY6d/4ms2+C1a/A0z+
E2w8Pr4xv/aWkpQZ+eS689SeI5sAn7t7yBaa3No+UqnWklCTtSH1teOGigXwKc3ZzNG1fbG+01KpXYOV8iGTDwZc
M7OcfeAEA0dUvs0dZPuAUPknanvqWGIUkd7PO7f/d9z6oOC+bv155An+75sTdX3qkvyRHSx2QF/aZyzc5UkrqLUM
pEWkJQ6mgoJ+mZQQRFladFd5/7g6Ihb+Urzx1ClUluSzd2Mx/z35/x69v3gTv/jW8XzwG6lfCbz8DP/9ouONS3Qr
43zPCnZdyS+CS3/Ns3gnc/8Iy2jviPZb3m3nLOfb7f+MvK7LDA07fk6QdaqoP/8BuCooIghRejd7eaKnNo6T2Pxu
KRg7s51dNC283LWLfivDPpcXrDaUk6YPqW2MUB8kQJgWhFEBOVoTcrBReKOlJcl2pKdE1gCOlpL4wlBJZbbXhbT8
uYLuy/EgSpeOgaT28fC8QXtX4fx8Jp9fd8eT7LN24gz/u3/gdxGPhYrNfeSlc2yMrD5Y8DXceCz85Ada/BcWj4TO
PsGHkEVx29wt84/evc+4dz/DOmobtvm1LeyfX/OZVvvbb11hZ28K9bye/XLXWQusA7gwoSUPY5ul7LZHC1H/41qO
lXvs1vPab8PgOFjjvV0UjoagKSHBE4VoAFjlSSpK2Ud+yZaHzVIVSg05ypNQeidWAa0pJfWEoJXJjYSCTV1zZb++
ZCLKIH/Xl8MHcH0FnuOveWQeO5ri9KmnviDPzoQXbXzT2tQfC24Mugux8mP4N+NILsO9ZkOiE5o1QuQ98bhZUT+U
Xzy6lLTlCasGqej562xz+e/b7dMa3vPd76xr52I+f4bfzV3Qde2N9J4mCZBDnaClJ2q62xjCUas8qSv2Hbz1a6qE
rINYElXvDhGNT8/lVBwBwQDTc5nvxhsbUfK4kDSH1TS3kB8k19zI1lEqOlBrRFn7XcKSU1HuGUpmus4P8ePhHdkF
Z/06FiB90MRRUQu0yWPB7INym+/pzDiA7GvD3d9bz1wVru79ow3uw4sVw16Op5285Xj4BPvl/8OkH4cSvw2cfhbJ
xtLR3cu9zYaB0/dkHcOp+o2jvjPP9R97mwp88y5INTfzh5ZWcffsc3lnbQGVRLj//7BFEIwENbR3ESsaH7++6UpK
0Xe1NteFtdpq+aGweLdXZFj4+7LOQqmmEVVMBmNixGIAlG/ySIUkf1Nq81YyDnDRcwBgMRuwJQFFT+J1iRU2LO7Z
KvWQolelaa7vulpb38/oc2flw1D+H9+fcAvFwNNPkkUVcfkJ4NeE7f1pAc3vHltdsHiU1+RQortr2PSefDCd9M9z
lAvjdSyvY1NTOuIp8LjlqAj/9h8O46eMHUpSbxbylNcy45SmufOAVmts7OXrPETz8L8dx4t4jmTiiAIDanDHh+zp
SSpK2K94SftmI56QplNp6tFRWHhz0ydR9djKUGtG0EIDFG5uIx/2SIUlba28K+4lYJBeyctJcTZokR0pFm9ZSFLT
S1hFnfWNbmouShgZDqUyX3HmvLlHAiJIBWC/k8M9DTjGsexMWPtZ1+Esn7cUeZfmsqmvlR0+8Fx6Mx+G1+8P7vfj
S0RlPcNec8Or1546dRDQSEAQBFxw+jkf+5XiO2rOC9s44QQBfOXkv7v38kYwqzgNg76rwy9WqYFT4ZjVL+qe9kjT
MJFpqAYjnlqaviGP/JZzCPeO7kN8/O8X2SnUYSuVufIvsKLR3xFlV5zohkrS12OZQKpqGtQcHi/wySC4LclhxLeC
6UlJvGUpluERzuKtSbaKIkUUDsKVqfhkcnlyQds7N4e5JQH5OlOvPDtfq+NnTi3hvXQMsfy6c6pdTDPuc2eNbP/7
WWhZvaKI0P5tPHDau23PjKgq47/NH8eOLP8Rv/+kYrpqxD9HIlukeU0aFQ4vfjyXX0XL6niRtX1u4eUSQV5K+GvL

LwyncR3whtZ87YgpEsgna6jm8LJzqvtjFziWpm87W+vA2O4NDKegaLXVQwQYAlm9yyrfUG4ZSGa61IQylaiiisni
Ahtse9SWI5sLy52Hp3K7Dp+5fxan7jaIjnuAffzmfDc/8Inxi/3Mgp6DHt/3pU4sAuOSo8RTmZm3zfCQS8JEDR3P
ohG2vqk9JjpR6vTl55d/pe5K0XVnJzTCiBWXpLSQdsnJg5L4AHJ3cgc9QSpK662wNL16kbZr3YJHcgW+f7PWAI6W
k3jKUynBNNesAaAhKKMjZNtjpF8VVcMinwvtP/2fXaCmAf/voAYwszmXl+hpy3vkjAK37n7+9d+lm/tIa5i2tISc
a4dKjJ/a5pM3T916oSV75r13WrS5JUig7Fo4Qys7EUAq6duA7MHvzDnyGUpLUTXJELZkeSiVHSk0IVgOGUlJvGUp
luJa6MMlvyRrgtUKO+QoEUXj/b/DXb3YFQOMqCnjsyhP45uQllATNrEyMYMaDnTzz3oadvt3Png5HSX3skDGMKsn
rczmTKguJRgIWtpWRIICOFmhc1/d2SdIwl9sRftnILapIcyVpklxXalJnuIahoZQkdRdp3zzNO8NDqeQOfFWxlQC
sqHH6ntQbhlIZrr0xnL4XyxngUKpiEpx5U3j/uTvgT1+BeCcA5YU5XFr4PABPZE1nWU0bn/rZ81z721dZ37DtrhV
LNzbx6II1AHz++D13qZycrAgTRxQQI4u2gurwoOtKSdI28uNhCJNfUpbeQtIlOVJq5OYd+AylJKlLIpEgSIZSWfl
pXHtwMEiOlCppWQY4UkrqLUOpDNfZFO6+15mXgt2MDv88nHMHBBF46Rfw4OXQGYOmDfDeLAA+/tmrufToCQD8et4
Kjv3B37j2t6/yzpqGrre5a85iEgmYvs/Irml4u2Lza2tyxoQHXFdKkrpJJBIUJsIQprB4RJqrSZOqaQDkNSwln1a
Wb2qmvSOe5qIkaXBoau+kIBGOCMopSOMurYNBck2pnNaNFNHMypoW4nGXB5F6YiiV6ZrDUIqCFE3LOORTcP7/QiQ
LXv8N/PpSeOU+iHfAmEMo3OMArj9nKr/5p6M5aFwZ7R1xfj1vBaff+hSfvut5Hn59Nb+eF67r8Y+7OEpqs82Lna9
gVHigZsluvZ8kDTf1rR0UE37ZKC7L0FCqaCQUjiIgwUE5q4gnYJk7KkkSADVN7RTSCkBWQYaPlMothsLwe8Xk6Fr
aO+Osb9x21oek7gylMly0rQaArMIUrhVywLnwyfvCHfne+QvM+tfw+IGf7Drl8IkV/OGLx/C7fz6aM6dVEwng6YU
b+OL/vURrLM4BY0o4evLufUHau6oIgIXtyfepXbJb7ydJw01dQyN5QQyA3KKy9BaTTsl1pY4pCqeOO4VPkkKbmto
pIjlNLTfD15SCrtFShxSGS6Qs9yKG1CNDqQyX0x5u9Z2T6mkZe58On/oNZBcCiXAR9Kkf73ZKEAQcOqGCOz51KLO
/dhKfPXYSRbnhDoFfPmkvgiDYrRKmjAo7ztebysIDTt+TpG4aajdueZCbwVfAk+tKHZS9AoAlhlKSBMCm5naKg82
hVAb3E5sl15XaPzfctMl1paSeZaW7AKVXfkcYSuWXjkz9h+95Inz6QfjtZ2Hfj4RTJHZgXEUB//rR/fnqaVNYW9/
KXqN2/0rMpMpCsiIBC9srIBcXOpekD2isD6d4N5NPQSSa5mrSKLmu1OR4uAPfIkMpSQLC6XsVm0dK5RSlt5jBILk
D3+ToWgCWbLS/kHpiKJXhiuL14W15VXoKGH8kfPUN6OWop+K8bIrzsvvlo3OyIkysLGT5uuSaUnUrobMDov7fQpI
AWhuSoVS0iII015JWyZFSVS3vAwkWb2hMbz2SNEhsampnXOD0vS7JkVJj46sAWLjO/kLqidP3MlmshTzaASirGJW
+OnZzGt7u2LuqiHWU0RHJgUQn1K9IWy2SNNi0NoTrDrZFM/zqd+XeEMkmu6ORscEG15SSpKSaZteU6ia5plR5a/i
d4r21hlJSTwylMlhL/XoAYokoI0ZUprma9NhrVDEJImzKqg4PuK6UJHWJNdWGt9kZ/kUjKwdG7gPAvsEy1ta30dT
WkeaiJCn9NjXFtlpTKsP7CoCKcPpeTnsNJTSyaEMjHZ3xNBclDW6GUhmsdsM6AOooojA3M6esbd6Bb3kiuZ5VzZL
0FSNJg0xnS214m+mhFEBVuAPfh3KTi527TogkJdeUaggfFKRwN+/BKqcQikcDsF/OemKdCZZsdAc+aWcMpTJYfU2
4AF9jpGi3d7IbqvauCr9oLWxP7j7oYueS1CXREm6GkXBHpa51pQ7OWQngFD5JApoa6ykI2sIHhWnYOGkwSq4rdWR
JOAX+vXUN6axGGvQMpTJYS124VWlztDTNlaTPxBHhDnyLOpKhlNP3JGmLtnAzDPIyt5/oUh2OlJqSWALAey5eK0k
EzeHMi85onrvvbZbcge+AvPC71kLXlZJ2ylAqg7XVh78o23LK0ltIGnXtwJdILvTuSClJ6hJpD6/uRvMNpaiaBkB
l+0ryaeXt1V75lqSgeSMAnfmVad28aFBJjpTaMxrOSnnXixjSThlKZbCOxmQnkluW3kLSbO+qoq3WlDKUkqTNcmJ
h8JJdWJbeQgaDopFQOIqABPsEK3hrTX26K5KktEokEuS2hhe5KczMTZO2K7kDX1UsXINw4VovYkg70++h1MyZMwm
CoNu/6urqrucTiQQzZ85kzJgx5OfnM336dBYsWNDfZagXEs2bwjv55ektJM2mjCreMlKqaR20uxihNNDsK4aG7I7
wD+m8YhevBbqm8O0XWcrSjc00ugOfNKDsKwa3+tYOygkD+mjxqDRXM4gkR0oVNS0DEixa3+QOfNJODMhIqQMOOID
Vq1d3/Xv99de7nrvpppu4+eabuf3223nxxReprq7mtNNOo6HBBDnVIq3h4ntBwYg0V5Jee1cVU0chTUFBeMApfFJ
K2FcMbk1tHRQmwsW8i0ozu5/oklzs/JDccLHzdxwtJQ04+4rBq6apnRGEG2IYSm2lYhIAkbY6Rmc3094ZZ9kmL3p
LOzIgoVRWVhbV1dVd/0aODKdGJRIJbr31Vr71rW9x3nnnMXXqVH7+85/T3NzMfffdNxClaCey22vD2+LM/rIxpao
ICFgWT3amTuGTUsK+YnDb0NhGMeEf0blFZektZrBIrit1UHY4JeNN15WSBpx9xeC1qbmdyiAZzrvz3hbZ+VAyFoB
jy8PQbqHrSkk7NCCh1MKFCxkzZgyTJk3ik5/8JIsWLQJg8eLFrFmzhhkzZnSdm5uby4knnsjcuXMHohTtRF4s/CW
ZX5LZodTmHfiWxpOdqSOlpJSwrxjc1je0UZIMpQJ33wslR0pN6FgMJHhrtSOlpIFmXzF41TS1UxmE3ycMpT4guQP
fhwrDNXxdV0rasaz+fsMjjzySX/ziF+y9996sXbuWxYsXM2PGDGbOnNnbm5ubOfHEE5k7dy5Dhw6lvb2dRYsW
2tq7H9fXhH4GxWIxYLNbfTRhQm+sdDHUXdtZDALnFI/q1nsHUxt4IgIkjClhaE46U6tzwPvFe1D7U2rmrMqGdmdB
GGFzts6/YucHwM7m6pon9k1fAYzml0M+1DIY29lnZJLIi2eR1NjE22MBbq8p6rH9ItnMXZEI7M6GNMLjaZ1+xc+n
+mVxf38K45JpSHXnlJAagjnS3cVdFyicTXfwU+0RXAtN4Z039TtswVNvZV5nQzkxoI/Rv+/o9lDrjjDO67k+bNo2
jjz6ayZMn8/Of/yyaKgOAD24UmEoltjm3txhtv5Prrr9/m+GOPPUZBQUE/VZ5as2bNSuvnt3fCdMLEfsHb7/P
yyv7/P02629gXRfEISxPhwpnr33mO5zse7vVrh1I7d0cmtHO4t7G5efCsZ2Bf0Tvp/Jl8YVUrHw3CL25/feZVOqN
vD8jnDLX/303Pqaa0dTn7Bst4amUlf/7Lw0R6sQv6UGvnrsqEdg73NtpXDD3p+pl8dlXAwcmLFy+8sZj1y3r/93N
fDbX/343fEHAIUL7hJeDDvPTeah5+eEWPrxtq7dxVmdDO4d7G/uwr+j2U+qDCwkKmTZvGwoUL+djHPsbixYtZ
6dNc569at2+Yqx9auu+46rrrqq7H9fX1jBs3jzz5Xv9qruec2
tIdfOXZQJ7RzubRzMbbOv2L501V7f0kZFcuZFdunoAfk+sdmQ++8zZhpEsom21bJn9iYWxUawpiHGxMrCnb5syLV

zF2VCO4d7G/uzbf2+0Pk111zD7NmzWbx4Mc8//zznn38+9fX1XHrppQRBwJVXXskNN9zAgw8+yBtvvMFll11GQUE
BF198cX+Xop2o37QegDZywx0iMtzeVUWsSowglohCZzvUr0x3SdKwZl8x+CUawvVa2vPd5rubqqkA7MMSABc7lwa
QfcXg1t6wgUiQIEEABZm9cdI2snJh1H4AnFy6GoB3Xexc2q5+Hym1YsUKLrroIjZs2MDIkSM56qijeO6555gwYQI
A1157LS0tLXzxi1+kpqaGI488kscee4zi4uL+LkU70VwbhlJN0RK2vVaUeSZWFhKJRFmWGMXkYDVsWgRl49NdljR
s2VcMflkt6wBIFO54GkxGqp4GQGX7Sgpp4c3VfsmQBop9xeCWaAy/T7TnlJEbHfAJOEPP6INgzWsckbuCnzGVhes
amdH/MxylIa/ff3vcf//9O30+CAJmzpzJzJkz+/uj1Qet9WEn0po9tObOD5TsaIRJlYUsqalmMslQas/p6S5LGrb
sKwa/vNawn4iUVKe5kkGmsBKKR0PDavYNlvHW6op0VyQNW/YVg1ukOewnOvMr01zJIDXmYHj5l+yTeB+AhY6Ukra
r36fvaWiINWwIb3PK0lvIILJ3VTFLE8kRAZsWpbcYSUqjRCJBUSzsJ3LL+28jjGEjOVpqv8gyVta2UN86vLd9lqT
tyW7dFN4pNJTartEHA1Dd9DaQYOG6xrSWIw1WhlIZKt5cE97mlae5ksFj/zElLOkKpRantxhJSqO6lhiViVoACkb
skd5iBqPkulKH5Ybbe7/tFD5JGaYzniA/Fu6+FSl27cHtqjoAgii5bRsZRS3vrWukM55Id1XSoGMolalawisbgYs
SdjlwbCnLDKUkifUNbYwKaoHkjkrqrjoMpQ7MWg642LmkzFPfEmMEdQBkl7j24HZl58PIfQA4JHsJbR1xVtQ0p7k
oafAxlMpQWW3hSKmsItfC2GzaHqVdI6USmxZBwisZkjJTGEqF/QRFftnYRvWBAIzrWEyEuKGUpIyzqbmdEYS/+6J
FjpTaoeQUvuMLw529F651Cp/0QYZSGSiRSJDTXgtAXolzwDcrK8ghUjaejkSEoKMFktuhS1Km2VhXT2mQvJprKLW
tij0hK5/seBsTgzWGUpIyTk1TOyOC5O8+15TasdEHAXBQ1lIA15WStsNQKgPVt3RQFA9/IZZW+GVja/uNrWRlItm
xuti5pAzVtHEVALEgG/Jde3AbkWi4Vgiwf7CUd9Y2uE6IpIxS0xyjcnMo5UipHUuGUhPb3wPcgU/aHkOpDLSitpm
yIPyFmF3kmlJbmza21B34JGW89powlGrKHgFBkOZqBqnkulLTspbRGouzZGNTmguSpNSpaWqnMrmmFIUj01vMYFY
9DQgobl/LCOocKSVth6FUBlpV20o5yV+IBa4ptbUD9yhlSaI6fGAoJSlDddaH05db8/yisUPJHfgOzQvXCXEKn6R
MsqnZ6Xu9klsElVMAmBpZwnvrGok7slbqxlAqA63c1ESVC9hu1wF7bBkp1b7+vTRXI0npEWlaC0BngX3EDiUXO58
SXwIYSknKLI31dRQEbeEDR0rt3OZ1paJLaIl1srK2Jc0FSYOLoVQGqtuwirwgRpwIlI5NdzmDSml+Nq3F4wFoW2c
oJSkz5bSsAyAoNpTaoar9ASjt2EAF9by12nVCJGWO9vrw4kUskgs5RWmuZpBLhlKH5y0HYOE6+wtpa4ZSGah9wxI
AmvOqIJqd3mIGoYLqvQHIqV8KCYfXSso8+e0bAMgqG53mSgax3OJwFz5gv8hSR0pJyiyN4cWLtlzXHuxRMpTan3B
pkHfXuq6UtDVDqQwU1C0DoL3IUVLbM3rCvsQTAbmdTdC8Md3lSFJKxTrjlHaEv/vyy/dIczWDXHJdqf2CZayua6W
2uT3NBUlSagTN4cWLjjw3TepRcrr3iNgaSmlkoaGU1I2hVAbKb1oBQFA+Ps2VDE77TxjFapILwLvYuaQMs6mpnVF
BLQAFIwyldqp6GgCH54X9qlP4JGWKrNbw4kXC9aR6ll8G5ZMAOCCyxOl70gcYSmWYto5OyttXA5BXOSnN1QxOB4w
p6VrsvGHVO2muRpJSa31DW1coFS2pTm8xg10ylDogGo5AdgqfpEyR1xaGUpEiQ6leSU7hmxosdgc+6QMMpTLMmrp
WxgbrAcgbOTG9xQxSxXnZbMoNpzZuWP52mquRpNTaUN/ECJLhSpGh1E4lp++Nji0jh5ihlKSM0NEZp6gj3Mk7u8Q
NMXolGUodGF1Cc3snq+rcgU/azFAqw6ysaWGPIJwDHpRPSHM1g1e8LBxF1rbWHfgkZZb6jauJBIlwh9bCynSXM7i
VjoW8MqKJTqYEK3lrjaGUpOGvriXGiCD8fZdbaijVK8lQ6uCspQAsWGV/IW1mKJVhVtY0MTYZSlHmmlI7Ulg9BYC
c+iXpLUSSUqx10yoAGrPKIBJNbzGDXRB0TeHbL7KUd9c20tEZT3NRkjSwaprbu0bURosNpXpl9MEAjI2voohmXll
em9ZypMHEUCrD1K5fRW4QC6+Al7iA7Y5UTdoPgIq2lWmuRJJSK1YXrjvYnOM6Ib2SnMJ3YNZy2jviLNrQlOaCJGl
gbWqKURnUhQ8cUds7hSOgdBwA+wdLeWVZbXrrkQYRQ6kM07Yh3E2uMbcKotlprmbwmjQl/JJRRgPr161JczWSlEK
NawGI5RtK9UpypNSHcsMd+PyiIWm429TU3jV9D3ff673kFL5pkcW8tqKWThc7lwBDqYwT1C4HoK1obJorGdwKi8v
YGJQDsHjhG2muRpJSJ6spDKXiRU7J6JXq8CLG5M4lQIJ5SzeltRxJGmi1TS1U0BA+MJTqvWQodVDWUpraO3l/fWO
aC5IGB0OpDJPTGF7JTZS6nlRP6vLC4G79Unfgk5Q58lrDdQejJaPTXMkQMXJfiGSR31nPGDYyb2lNuiuSpAHVXLs
+uSFGAAUj0l3O0JEMpQ7JDhc7d2StFDKUyiCJRILi1nAB29zKiektZgjoLA934GtftzDNlUhS6hTFwlAqt9xQqle
ycqFyHyBc7HzR+iY2NbWnuShJGjixunBpi9asUohmpbmaISS52PkeHSvIp5WXXexcAgylMsqGxnbGJNYBUFi1Z5q
rGfwKqvcGILtuCYmEc74lDX8t7Z1UJMKRPoUjnObda8kpfMcVh1/UXnK0lKRhLN64HoDWnIo0VzLEFFdBUTUR4hw
QLHEHPinJUCqDrKptYWwQdiJZFRPSXM3gN3JceOW7qnMVa+vb0lyNJA28DY1tjAxqAcivGJPeYoaS5GLnhyUXO3c
Kn6ThLGgOv0/E8t15r8/GHQHA4ZF3eWdNPc3tHWkuSEo/Q6kMsqqmiT2CcFoGZa4p1ZOcUXsBMCFYx2sratNbjCS
lwLr6VkZSC0BQXJ3eYoaSqnCk1MSOxYAjpSQNb9ktGwGIFxhK9dmEYwA4Pudd4gl4fUVdmguS0s9QKoPUrFtObtB
BJxEo2SPd5Qx+FeGaUqOCWt5ZtjrNxUjSwKvbtJbcIHnV1t33ei85UqqoeTmFtPDqilraO+JpLkqSBkZuexhKRYr
rcea/PkqHUIbxNhLhT+CQMpTJKy/olADTkjHJRwt7IL6c1uwyAdcvcgU/S8Ne0cSUAjZHicAFv9U5hJRSPISDB0QU
raOuI88Yqr35LGp7yY+Fo0KySUWmuZAiqmgq5JeQnmtkvWGooJWEolVESNeH2o62FLl7bW51l4Wip1rXvudi5pGG
vvTYcFdqU45SMPpt4LADnFr8DwPwlTuGTNPzEOuOUdNYCkFfqLq19FonC+KMAOCLytqGUhKFURsluCBdgjZe6nlR
v5VWF60qNaFvBytqWNFcjSQMrXh+GUq25Tsnos71OBeCIzpcAmO+6UpKGoZrmdkYG4UjQvHKnee+S8UcDcGTkbVb
XtbK2vjXNBUnpZSiVQYpbVwGQPWJiegsZQqIjJgMwIVjLs+9vTHM1kjSwIk1rAegsdEpGn00+GYCRjW9TSR3zltY
4wlbSsFPTFGME9QBEi+wrdsmEcGTtUVnvAAleXlab1nKkdDOUyhDN7R1UdoRfNoqq90xzNUNIcrHzicFa/vb2ujQ
XI0kDK7sl3KE1KPbqd58VjYLRBwNwctbrbGhsY9mm5vTWJEn9bFNTOyOCMJSi0Kneu2TMIZCVR1minsnBKqfwKeM
ZSmWIVbWtjA3WA5A/clKaqxlCKsIAb3xkLU+9u562js40FyRJA6egLQylsl0nZNckp/CdVbgAcAqfpOGnvr6WgqA
tfOCo2l2TlQNjDwc2rytlX6HMZiiVIVbWNLFHEH7ZoMw1pXotGUrtEWyko72FFxZvSnNBkjQwEokEJZ3hNOW8ij3
SXM0QlQylDut4mQhx5hlKSRpmWmrWANAW5EJOYZqrGcKSU/iOiLzN6yvq6Iw73VuZy1AqQ9SsXUZu0EEnESgek+5
yho6CEZBbAoTrSj3xllP4JA1P9a0dVCbCEKW40l1ad8nYwyG3lILOeg4MFrkDn6RhJ1YfLgfSlFUOQZDmaoawCeF
i50dF3qapvYP31jWmuSApfQylMkTLusUA1OdUQTQrzdUMIUEA1QcCcELkNZ54e60L10oaltY3tDEqqAUgt8yLF7s
kmgWTpwNwYuRV3l3XQH1LLL01SVI/6mwIlwNpzalIcyVD3NjDIZLF6GAjY4MNvLqiLt0VSWljKJUhOjctBaC5wCk

ZfTb1XAA+ljWX5ZtaWOiVDEnD0MZNmygKkttSu9D5rktO4Ts993USCXjFLxqShpGgKZw1EMsbkeZKhricwnDBc+D
w4G1DKWU0Q6kMkVW/HIDOEqdk9Nn+50Iki6nBYiYHK53CJ2lYaty4AoDWIA9yi9NczRCWDKX2jS+kjAbmL61Nbz2
S1I+iyV1a4wXuvLfbJhwDhOtKGUopkxlKZYiClpUARCsmpreQoahwBEw+GYCzo8/yxFtr01yQJPW/1k2rAKjP8ur
3bikZA6MOIEKCEyKv87JbfUsaRnLawk1/gqKRaa5kGNhqsfOF6xppc5NvZShDqQzQGU9QEQt3yiis2jPN1QxR0z4
BwEcjc3lp2SY2NbWnuSBJ6l8ddasBaMn16vdu2+sUAE6MvsqrK+rodClCScNEQSwMpaIlTvPebeOOBAImR1YzIlH
LskYXjldmMpTKAOsaWtmDcFHC4urJaa5miNrnTMjKZ8/IGg5gMU++4xQ+ScNMY3jxoj1/VJoLGQamnAbAidHXaGm
PsaopzfVIUj8p7gh3Fc0zlNp9+WVQNRWAwyNvs9Rla5WhDKUywKqaJvYIwvnf0fIJaa5miMotgn3PBOCc6DM88ba
hlKThJSu5eG2iyC8au23cUZBdSCV17B8sZXGDV78lDX2tsU7KEuHaR/kVo9NczTCRXFfq8Mg7LHWklDKUoVQG2Lh
mOTlBJx1EodgOZJdtnsIXfZY576ylvSOe5oIkqf/ktYUjaqMl1WmuZBjIyoE9TwTgxMirLDKUkjQM1DbHGBHUA1B
Y7gWMfpEMpY6MvM3ShoBEwvneyjyGUhmgae0iAOqyR0E0K83VDGGTTyGRV0ZVUMv+sdeZv6wm3RVJUr8pim0EILf
Mixf9IrkL34nR11jkFw1Jw8CmhhYqaAAgcFRt/0iGUvsGy0jEmnhjVX2aC5JSz1AqA3RuXAJAU/6Y9BYy1GXlEOx
/DgDnRJ7h7+9sSHNBktQ/OuMJyjrDoL1oxNg0VzNMJBc7PzR4l0R7M68sd7tvSUNbU81aIkGCOAHkV6S7nOGhaBS
M2ItIkOCwyLs8+MrqdFckpZyhVAaI1C8DIFY8Ls2VDAPJKXxnRl/g6bdW4YVvScPBpqZ2RgbJUKpyjzRXM0yUT4Q
RU8gK4hwTWcCv569Md0WStFuaa8MNMRojJc6+6E/J0VJHRN7hz6+tdokQZRxDqQxQ0Bz+IRytcJHz3TbhGOLFoyk
JmplU9yzrWtNdkCTtvvW1DVQE4bY/WaWOqu03yV34pkde4S+vr6ahNZbmgiRp17XXrQWgMVqW3kKGmwnHAnB09C1
qmmPMfnd9mguSUstQKgOUtYdXNfJHTkpzJcNAJEpk6scBOCc6lwU1Ll4raeir3xBevIiRBQVOyeg3ySl8J2e9Sns
sxp9fc1qGpKGrsz7cpbUlZ0SaKxlmkiOlpgaLGBes5fcvrUhzQVJqGUoNc/WtMarjYQdSOmZymqsZJpJT+E6NzGf
RprY0FyNJu695YxhK1UXLITBs7zcTjiORX8Eoargw+iT3v7g83RVJ0q5rDkfwtOd58aJflY0nvudJRIlzZdbveeK
tddQ2t6e7KillDKWGuVWbGhkThAty543cM83VDBOjDyJWPpm8IMaUpnksXNeY7ookabe01awCoCm7Ms2VDDPZecS
PvwaAr2b9loXL1/D2GndWkjQ0xerD6XvxfPuK/hY/8ToAPhZ9hnHx5Y6sVUYxlBrmNqxeRk7QSQdRKHab734RBGQ
fdCEAZ0ef5T8fW5jmgiRp98Qbwmnebfkj01zJ8BP/0GU05oxiZFDHP2b9mQccLSVpCHp/fSMtNWFQUjVmfJqrGX4
SYz7E6tJDiRLnqqzfOIVPGcVQaphrXPs+ADVZoyASTXM1w8i08wE4IfIaTe8+xfOLNqa5IEnadS2bwul7nQVVaa5
kGIrm8NaYCwD4x+hfePqlN2iNdaa5KEnqm1/+/VVOj7wIQOWkA9NczfD01uiPkyDgI9EXaFv+Mos3NKW7JCklDKW
GufaNSwBoyHc3pX41YjLxgz5FJEjww5w7ufUv80kkEumuSpL67Kl31xOvDUOpMWMnpreYYWpV2eHExxxGQdDGZ2O
/4rE316a7JEnqtXX1rYx8/aeUBs20lE2Bfc5Md0nDUkP+WBLJDZWuyfo1DzpaShnCUGoY64wnqF+9CIBY0dg0VzP
8dJ72XRqyRzI22MDH193Gw6+vSXdJktQn8XiC//3zbM6KPgdA6cSD01vQcBUExE+9HoALo08yZ+5T6a1Hkvrgvid
f4tLIIwDkz/hXZ18MoM7jryUeRDkp+ioL5z1OPO5Fbw1/hlLD2N3PLGZkw1sAjJm4T5qrGYZyi3ll4uXEiXB+9Cm
e/8v/0t4RT3dVktRrf3xlBZfW/IiCoI3YuGNg34+ku6RhKzHuSJonn0E0SDBj1X+zbGNzukuSpB7VtcQom/9jioJ
W6sunwn4fTXdJw1vFnsQP+hQAl7b+khcXu0SIhj9DqWFq0fpGFj72P5wenUeCgOL9Z6S7pGFpU9HetB95BQBfbb2
DPzw9P80VSVLvtMY6efmRuzkp+iqdQTbZ5/wIgiDdZQ1rBWd8l04inBp9mWef+EO6y5GkHv1h9gtcxF8BKDpjpv1
ECmSd9HU6gmyOirzF60//Md3lSAPOUGoY6own+PGv/sD1kZ+FB078Oow7PL1FDWPRk77OxpL9KA8a2WP2NTS0tKe
7JEnq0QNPvc4V7WE/ET/uKqickuaKMkDlXizf85MAHPjWf9LR0ZHmgiRpx1pjnRQ9fyu5QYz1FYcSmXJqukvKDKV
jWb/vJQActfjHtLbbV2h4M5Qahu6b/RpXbPgOeUGM1oknE5z49XSXNLxFcyi5+H9pI4djeYXnf/0f6a5Iknaqrjl
G4dP/zsigjvqiSWSfeHW6S8oYo8/5NxrJZ7/E+7zz2E/TXY4k7dBf5zzH2fEnACj/6HccJZVCVWd+k2bymBq8z2t
P/F+6y5EGVFpDqTvuuINJkyaRl5fHoYceytNPP53OcoaFxesb2OPJrzIxspbG/DHkXXAXRMweB1p29f4sOuhrABy
76FY2LH4tzRVJw4d9Rf/7459+x/k8DkDhx2+HrNw0V5Q5ckureXHMPwCw3wvXsfKXl5NorklzVdLQZ1/RvzrjCfK
euYnsoJMVI44ha9Jx6S4po0SKR/HaHuHI2oOev5p377+ORLtrEWp4Slta8cADD3DllVfyrW99i5dffpnjjz+eM84
4g2XLlqWrpCEvHk/w7M+/xcnBfNrJpvCS+6CgIt1lZYx9z7maV3IOIT9op+jnp7Lo7i/QuvKNdJclDWn2Ff1vxYZ
ajn7z3wFYtecniPpFI+X2+th1/DE4mQgJ9nj/fur/8yCW/f1/IeEuS9KusK/of888O4fTYrMBGPHR76S5msy057n
f5tnIIeQSY++372DdDw7h/Tm/TXdZUr9LWyh1880387nPfY7Pf/7z7Lffftx6662MGzeOO++8M10lDXmP/fl+Lmz
4BQCNJ99IsMchaa4oswSRKNHz/pvXE3uSRxt7Lv01eT89loX/cTLvPf0AiU7ng0t9ZV/R/1574N+ZEqygLlLG6PN
vSnc5GWncqHKmf+3X3LvfnSxMjKU0Xsf42V9l4U3TWf3eq+kuTxpy7Cv6VyKRIDr7RiJBgoUjTiJ/omvTpsOoyko
O+vos/rLvD1idGEFV5xomP/45Xv/PM1mz9J10lyf1m6x0fGh7ezvz58/nG9/4RrfjM2bMYO7cub1+n1cfv4+iwvz
+Lm9AdXbGaVu+iFdnbSIa7TkTTCQSXVdOt1xAjdPZ2kxHSx3x1nporSdob+DI2tlEgwTv7XEue53whYFrhHZo2r7
7svLKZ/jt3/9E5YK7Ob7jOaY0zYcn/pE1f/sW63InEssppSO3jERuKRSUE80rgUiUIBKBIAqRCEEQASJbpu5vNYc
/SPF8/r7+zA5FmdBGgLqGpnSX0Cf2FbvfV3S0NNLRtCmcHtZaS1Z7Hac0PQMB1J1wPaWOpk2b0oJsLrnwYlZu/Ah
/fuC7nLL2Hqa0vELHL6ezIjKKhqwRtOSOIJY/ikRhFZGiCoJIFkEkCpEoBFGCaBSI2lekSCa0EewrhpJd7Su27id
IJIg11RJrWEeiaQPZLRvJa9/IsZ2vE08EjDrbUVLpVJCbzUc++U+s2/AJ/v6rb3Hchl8zrfEZ2v73WBZFx9CUVU5
bbgUdeZUkikYSLaiASFbyu8WWfwT2FamSCW2E/u0r0hJKbdiwgc7OTqqqqrodr6qqYs2aNduc39bWRltbW9fjuro
6ACbN+RoluUNvwb29AVYOzHu/ljWJ0R/5dzZu3DgwH9ALsViM5uZmNm7cSHZ2dtrqGGg7amcecNKJpxI//hSeevs
tGp69m4M3PUJZsI6JrevSV/BuGMif2cEiE9pY37Y5tBgaU4TsKxiQn8k24NX8DzF56ln2FSnQUzvzgKMv/AZvvHc
OrQ9/m4Pa51PCakpYDY2pr3d3ZMLv0Uxoo33F0DJQP5P1wOsVp7Fv0R72FSnQUzujARx48fd45e0L6Pzr/2P/2Bt
UsoRKlkBD6uvdHZnwezQT2tiffUVaQqnNPpjMJhKJ7aa1N954I9dff/02x8fdMsT+WkuJ1+C749NdhKRBbOPGjZS
Wlqa7jF6zrxgIs+HfRqa7CEmDmH2F4Hdw7e/SXYSkQaw/+oq0hFL/v737j6mq/MA/r+/Ag+Ea4iZ+a4EaeyA

MAEnJ3xhTyZxeAt1MbTTMnG1O59w0v6mVLsxStGZam+Ka+YslPxRHiuJEcwYKqQVRkMkAIeH5/sE8er3QFYF7OM9
9v7a7cc85HJ4Pn2f3vT3n3Hs9PT3h4OBgcfWisrLS4ioHAKxcuRKpqanK8+bmZlRXV8PDw8Pmtxx21N27d+Hv749
bt27Bzc1N7eF0CXuoEWCdMrGHGoGWq8EBAQHo108bb9liVsg9J+2hRoB1ysQeagSYFVpiD3PSHmoEWKdM7KFGoHO
zQpVFKScnJ4wcORKZmZl4/fXXle2ZmZmIi4uzOF6v10OvN0ON/+66j59+nT1MLuUm5ub1JMUsI8aAdYpE3uoEQB69ND
G+9uZFfYxJ+2hRoB1ysQeagSYFVpiD3PSHmoEWKdM7KFGoHOyQrW376WmpmLevHkIDw9HREQEQEduzYgZs3b2LRokV
qDYmIiLoZZgUREVnDrCAi0i7VFqUSEhJQVVWFtWvxory8AgiIrKGWUFEpF2qftD
54sWLsXjxYjWHYN6vR6rV6+2eNuKvLZ0jIIcjVuyJZ5b9Y/QqINQAHsw3RdKtZgmjxOgUjrSP7XKxxNbqB6fTmm
PSyzYQSPcE0TIHcA+UCsJgxEarRV/zrwpr8vgs2m9vZ3+p3YNrRasFlt2pu3oU8VpW3UlhTYUbkm9dSnnsB94Z2Eez
WqRNa+b5XIiIiIiIiKShjY+wZCIiIiIiKTCRSkiIiIiIiIrI5LkoREREREREZHNcVHKBtasWQOdTmf
28PX1VXtYHZabm4u+tp06vRcnMMa4UahkdfvFg2pkcy0oyD9HKqSDUjSEzVD9M7pBDmVL/AH0yVftYhMyxrUc36OmgaRm6O46sdFGVzEfwYjP6M2gvj6L7Ib5
I0c+uzAouStnA6NGj8dVXX+H48eP44osvUFFRgTFjxqCqkrtoXWZiooKAICPj4/Zdh8fH2WfLCZNmoS9e/fi5Mm
T2LRpE86dO4eYmBg0NDSoPbSnSnIoRAamoqxo4di5CQEAy9rO1OgE5+llUVASDwQCKC9Xo9FixYhIyMDzz
z4iEZ56QMry2PYlZov5/MCm1iVjwk45yU4bXlUcwK7ffTFlnRs9NGS22aNGmS8S8vOwVgGqkm3ZfUzaIPF3WhJ
xZF1Pp9OZPRdCWGzTuoSEBZRsc6BQ6R2ySYFjdYHDR7pPEHIzAwED/88APi4+NVHNnNnTWbp0OKUM/Bw8ejIs
XL+LOnTs4cOAATCYRnJylRHIkPtLvMNZaBYOmyBAmoFKz8/ylbA+VjSDrzW/lhFwFfSpj3lrRhJwAsCPbPWIUt6
GDRuG4uJit6/ngq+tycEBiPrXggj2++eRFqNwvk7++uiT2vVHV0t2K5A9+Fh6Kuc5KLJamGb5mN3QHIB9Ur1d46i+C4hU60yWqhYPIg
Y4eHhWL9+PYYYPH45PPvlEuj7Kjlkh95zU4mvLA8wKS1rsJ7NCDswKueekVV0tb1JNRc/BTv/+0Bi5rAeo9PXD8tlZ+P2Ft0
Ko9Go9lC6TFBQEvDx4ECdPnkRQUJDx9fZZmalsa2xsRE5ODsaMGaNiZHdXcHAwfH19Tddcb0bxuXKFmuqqtdOJzsaSrBzGn9+nP6KnXQvDOwC+vvZ6fsuHKLmQeUxj91l9jmbpn2lvp
nGvTp5twnJGWCT4I9+uS3zAVcbuzHtvzqbfJH2HYNBnjHN5CPJaoPS6uRdhc2/0GTa3mFBSeaYb5WcqKXS9r9sNJmUnKvB11WA4UAJBkX4b3ah2DXTT86w3Fae6ivZtKGR6Ty51
WqRNa+b5XIiIiIiIiKShjY+wZCIiIiIiIiKTCRSkiIiIiIiIrI5LkoREREREREZHNcVHKBtasWQOdTmf
xgOgd4ohcyNgRhjq3VD/RAxQCvJ4dmKxlxl1n7vVCyxuHIyACb1Zt/VIWObIAGJRr99UkO/FbzsF8KwI3/HNd8lt/KbgF9E1S2a6XK1Ycv7dxPSdYOk2RdhpZmyjGMjB4XXzXrw7MZqIUvX0HAaiiUx
dG2FH/7qM3vAjbDo+ZGuAQsjp7lg41dqvzZAYvr5KSVG+6MaeafRHRlUUxQzVaH7Lk4YTAo6sb2LSHtVIUq5uZEiMqprmZMnGGE9zUrBfFnRL/A+wBrzq3uPNbxj43gxcVXr2uxv2KYm55X6ptOrlSHqP1eeuNN1bszr7h44XXOk+CRNkFYMWWjcz2cCqFU7Er7LKe3

v4e3t/VTvswaAYcOGobm5GTk5ORZXGh6YPHkyXFxckJaWhqNHjyI3N/c/zxkWFoa0tDQIIaDT6do9pgMHDmDt2rU
4evRoq+8vf3DFJiwsrN3nJiLqLpgVzAoiImuYFcwKUhc/6Jw0acmSJaiursacOXOUb3M4ceIEkpKS0NTUBIPBgOT
kZCxfvhw//vgjLl++jMTERItbSWNiYrBt2zZcuHABBQUFWLRokdnViblz58LT0xNxcXE4deoUSktLkZOTg5SUFNy
+ffuJxjpgwACYTCYkJSXh0KFDKC0tRXZ2Nr777jvlGAcHByQmJmLlypUIDg42uz24NePGjUNtbS2uXLnSjv9ai8u
XL2P+/PlYsWIFhg4dioqKClRUVKC6ulo5Jj8/H3q93uo4iIi6M2YFs4KIyBpmBbOC1MVFKdIkPz8/5OXloampCa+
99hpCQkKQkpICd3d3JSD+97//ITIyEtOnT8f48eMxduxYjBw50uw8mzZtgr+/PyIjI/Hmm2/ivffeM7u9u9tXfv3sj
NzUVAQADi4+Px3HPPISkpCfX19e26wpGWloaZM2di8eLFGDJkCBYuXIja2lqzY5KTk9HY2IikpCSr5/Pw8EB8fHy
rt/9aU1BQgLq6Oqxbtw5Go1F5xMfHK8fs27cPPc+f/O/c9bfYmIujtmBbOCiMgaZgWzgtSlE4+/8ZVIYtHR0QgNDcX
HH3+s9lAs5OXlITo6Grdv34aPj4/V44uKijB+/PhWPzCxI/78Zo8UdFmlV7ZsUjWcnDv2VxxTGCdZUbT/lmIeAfMMuFBkMnPPmm/A8KPmd3Q5NoU5P25VQQbRmX4k1b3c2SZOh3TWk0k8lUtRVF7mXYmzJbDmCPUPzg
1zAp6ErxTikhlDQ0NKCkpwapVqzB79uwnCg6g5T3lGzduRFWaeOp7S0FJ999hmDg4ioG2FWEBGRNcwK0iJ+0Dm
Ryvbt24fk5GSEhobi66+/btfvmkyTh/PqFGjMGrUqE4/LxERPT1mBRERWcOsIC3i2/eIiIiMjm+PY9IiI
iIiIiIiKyOS5KEREREREREERGRzXFRioiIiIiIbI6LUkREREREREREZHNclCIiIiIiIiIpvjohQRERERERERERERQ2x0UpIiIiIiIiKyuf8Da0Jun6gTOJ0AAAAASUVORK5CYII=",
"text/plain": [
"<Figure size 1200x500 with 3 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"psd_r, freqs = psd(trials_filt[1])\n",
"psd_f, freqs = psd(trials_filt[2])\n",
"trials_PSD = {1: psd_r, 2: psd_f}\n",
"\n",
"plot_psd(\n",
"   trials_PSD,\n",
"   freqs,\n",
"   [channel_names.index(ch) for ch in ['C3', 'Cz', 'C4']],\n",
"   chan_lab=['left', 'center', 'right'],\n",
"   maxy=300\n",
")"
]
},
{
"cell_type": "code",
"execution_count": 23,
"metadata": {},
"outputs": [],
"source": [
"# Calculate the log(var) of the trials\n",
"def logvar(trials):\n",
"   '''\n",
"   Calculate the log-var of each channel.\n",
"   \n",
"   Parameters\n",
"   ----------\n",
"   trials : 3d-array (channels x samples x trials)\n",
"   The EEG signal.\n",
"   \n",
"   Returns\n",
"   -------\n",
"   logvar - 2d-array (channels x trials)\n",

```
"        For each channel the logvar of the signal\n",
"    '''\n",
"    return np.log(np.var(trials, axis=1))"
]
},
{
"cell_type": "code",
"execution_count": 24,
"metadata": {},
"outputs": [],
"source": [
"# Apply the function\n",
"trials_logvar = {1: logvar(trials_filt[1]),\n",
"                 2: logvar(trials_filt[2])}"
]
},
{
"cell_type": "code",
"execution_count": 25,
"metadata": {},
"outputs": [],
"source": [
"def plot_logvar(trials):\n",
"    '''\n",
"    Plots the log-var of each channel/component.\n",
"    arguments:\n",
"              trials  -  Dictionary  containing  the  trials  (log-vars  x  trials)  for  2
classes.\n",
"    '''\n",
"    plt.figure(figsize=(12,5))\n",
"    \n",
"    x0 = np.arange(nchannels)\n",
"    x1 = np.arange(nchannels) + 0.4\n",
"\n",
"    y0 = np.mean(trials[1], axis=1)\n",
"    y1 = np.mean(trials[2], axis=1)\n",
"\n",
"    plt.bar(x0, y0, width=0.5, color='b')\n",
"    plt.bar(x1, y1, width=0.4, color='r')\n",
"\n",
"    plt.xlim(-0.5, nchannels+0.5)\n",
"\n",
"    plt.gca().yaxis.grid(True)\n",
"    plt.title('log-var of each channel/component')\n",
"    plt.xlabel('channels/components')\n",
"    plt.ylabel('log-var')\n",
"    plt.legend(cl_lab)"
]
},
{
"cell_type": "code",
"execution_count": 26,
"metadata": {},
"outputs": [
```

{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAA9wAAAHUCAYAAADInCBZAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABGyUlEQVR
4nO3dd3iUZfr28XNISCUJAFCIITQQiwUoSAgEgv4q4NLJR9AQUFRJdigYAURcWyqxQLZRVkVxAREUGRiAJKi6h
gAAlNAhGQhBpS7vcPN/NzTCEk82QyyfdzHDkO5mnnX9czcGXLOU8ZmjDECAAAAAABOVc7VDQAAAAAAUBoRuAEAAAA
AsACBGwAAAAACxC4AQAAAACwAIEbAAAAAALELgBAAAAALAAgRsAAAAAAsQuAEAAAAAsACBGwAAAAACxC4AcD
NLFq0SDabTYcPH3Z1KyXa4cOH1bt3bwUHB9tms2ns2LGubvt3bwUHB8tms2ns2LGubslB9uu4Y8cOy2rExsbBKZrPp9OnTltUoTvmN/T179sh
ms2n37t3F31gZs3btWsXGxrq6DLAsxOKFSsUFRWl1auLq/t2rV67bXXCN0AUAAEbgBAiXL58mX5+PjZGfk
KeaXN21mX9+PjIzCHZs9NyeaPzpWhFnD8F+1+RQrXJ6/kM8KJkwV+/6aR2GWTLTi+hx6D8kXK8hM0dxUxLb7
D0j8Ed0ZxNP1QuT4f7G1vbLoqkmdUiVf0g9Vg1QYNJiOJs0qUmBvdvdo/GSKE/9gOB2WHLsMmi55sUqmVS+3W6
sPDCXFwTOpk/0JOyhJeNVacnxhGwWrh78Rbx+fLKGyr3Fs+/8+5BlgXGSh8cZHBgS3rsfmtVFLSDj+gzZBeNfjh
eDAN2hIX2S3tGNZ2bc4gDCi5aLbntV9gfUx8LEWa7sOeZYh14iJkmXPTPUiTnTEWp8JVP9gkxz/lRkBdNgyXeKtPP
PeNQPIkG9gkYfNsT0kr4RLDDuMg1ONmQNDgpLdqaDYQGLX23MAGkLOfx9f9p7nUf0aAnE+x8F7yr2wYwHoKpPnJnJz
hO3xRUJT+yy1iIsx4bEuzYN3Vp3sEinb1VRsrMsmNrqxU8zUUVfTDhvIw1Wvg3wmjvkmVVHJ9qwtMPfJq0mPvLR8
sLrrPMVnyH/cn43/7/hOHuSUrgzHhtIbYTX8Qmn/KvsOFXn5dP1rgFXtQe4qPpBaHHfdGfR7xGPAzbEXYPAMvnvvG
YsE6f3r/51C41wIwWuaUXvOrQsoRZ1WeVnFa1dnhHtIzjyV/HzvPxxkN8YNb6hg+dxQpP2v8NOpZEc4U/WOUeeR
mA7ytjt/hnL6LmwR6i0KQjRf2GFcR3CC2kS/Aeak1vj1rsTFlsRGaMfnpbRN/Oxk/ahVzqt21n88Aq3NswhYQUtJq
AsCuNemc9QuaxRmaG6L4yCXOaTx8sEq6xgeiZBvtuRTeSGazYqe8YaRdrOJ4UGg+DOMQ7Z0xqtPLSiCnDEX26xHEos
Ul1jAkSSP+1JJVDOiWRqIWWsm9BJZabYrv42iOLh5MTq9sI/uvOzaBQfohGZvDqL2j6gNgj7eSljy5vC0Hm3zBs1dn
iP3OM5tRuRnVrOWcpsCw71kFEhS3/Xo3cXw4Q9dd3F/HEHp6VQ2xUd8GKbWHR2hTIhkgdPVTE6VGJm8cXXkhsO0l
ZiOLoikk3OX6FnQgAN/V5u7zf85zn3S6DUk59HcPOeDk4/ShTEuBsdKf05bs0KIkhIcshHdbR2GH3PLGjiWanb0n
0XHZzC9EQlOTIzHgUAYBcXD+CuWpB1zvQJ0W8aGrvd4+2wa6PVnFCmN4ZeNLLtKdgUF15LdbSf7LeSz84/lwn/H/G
ptBStH4R/0/pKvjbaw+CD5BSkxX9wOJXsRLAUM2jEeNzI/f5nPdLNCBIQ3yhzEt8SoE2RXgsO5G7+LKzjoo0FHcXf
Z49rGNzPjnVORIFozxx3AuEuhCSOOijJg7GGyzWuYCP1Rdr5CMNUXDFkq64tRkw79N7I/T/tr1IFR8mNsc+Iky+t
FGtKEWNbNVL6iwevdvM4JsbcsgK/YGUdvvx+O/iNSwjNzlZNVDu2TS3o+UCzQNgRFdIUXLMUR5lEjjfIHchANZGQ
BsRwBqHaOn3K+wRYbzqnc6/9XAhVSpAzNFXJN0Nsa6Jlmxqy3fjB+fTnIUgN4WpkM3DMLNK0ZKEN12N46L22Tzb
Rlaxx1pKpXmeLn7swp6ExEzDZ34hx9oNxCzjtObJfGsA82e2X3UcIoZ4XX6eL3q5+QbNPjeFNZx0Yyg8HXdHTvkT
yzJgjmGkEWFvHEvkK4kVaO5AOzyzAPimOtPz/mWZu67cS+LL1/IrCXn0rDa9n5Z4JJlfCI0MTgZCeEQJmqU0FhkG
dnMQ6U4zBIzATGqX2zVgs99IkjJ0QyBmyxtZtMUU4qTAybtmcukE+RDXr35MrkWgrkU7OTaAobDT38wxGXB+Cg/FC
Dn3y+oWi3xQ0Ubg/ejlKzwWh3e5WgegnzTHoMCxfi9zvvvCRMQE3HwMfd19Ds2MrOYSFk8xmmJ7aQyG2Ckt6DZy4
pJx4a4hwlKt1bbbiZ77eEOhf4QlmRrmeq7tzh3G3tKt7KAmlZIBq2wlP/mYUw6ZTz9GAPUU6LXQW2nAm/ah0afPh
NNBz6PiYmC07A6qL9pnqXQkLXLabTkEhUfCDjfO6xe35dPnd83yLDm7emzrI2d2CuNyN3hKzudGIoFBqBCgVOE
oFFHajqGAY4FLwG6NpW5Dwj5vuZlMFmwvvS0cppoc9tr4OvYWRiC7yxGIK+y2uxZ1MzBH+ZejNlMg3cEGsDBLPO
V6i2bOxXkMZXcJgwNSMl+c2yqjuSc7o1RmKvxSSAmsJ3rhsl7FYPtwL4Gj0r9sGKoMwEscj+eaCtzZy/Om1NgZOw
5ZkaDLIs0jZbtXAyZRtgUo0HN7+J3cr1ZFJvnTlt9KCDjVTfjmrjetSwN2rgC6NcVHHYE0dOFMUP/V1Fm8dPXvFZ
fsZfU8v1Vw4vX7ibaUXOVVSUsyV8Kt0vbfLNEyPVFzK9LlJfGvcX2XFHWqnwfG/uOQkPbgMxZc5vJHzNwtWlLVbAL
bjr65PhAoJHfeuWaC0dXAX3y4LfYYa8ZY7V7Kk82OXQxubEWnt1s19xK3r/Wdd4vbT0c1rKr81rSmZpJyjmI81n
M6zyk9fVX7pVOYGaFTuKUmsO5bYjW4YQF9t5rLTG7ctRq4S5pNhrVxHYd5Zx3y3dyo9HhlwdLa39ZIzThMPGlt9v
HRWe8WCKgD44A9UWNGGe9W7JfrpmCCY8ZVdOvqsSyc8BMfcO5lnfqrZQybSk+bSq3x5BTelgmCMovq9pw+cnU6Xd
6Lq3gywUBPtKoU7Swx1tzWBnE+UKoLrbd/LDdlzmoKr7RRy24sMBbbw/aD7l+TU6YX06u42kdIKbvXZhaScy
QWZaalDRXY2e8gdm8NUW8AfKwjaAtPJ82R4xWmQlSZGmqjX5AR2iMnlGt1npe0U5hKDXKMXeszeQuW6SzZczaJIzq
AFqdWq2EBEb/9O+wy4IWPs04ezK3eRUS7IiPdG27W4zWGE3lrZ4Gc7dQt/M2LZ/74fAEKIlY7U1Ex7b8oLOkFIJCNY
WPfDc5Rzv4QW2ZvTGZyAw3/Z6TtFukw3Ufqf1tWzrz25Tt9E0bB/vWyG9cx+N3Pe+Hd81XEpx66qNdMZPX5KMnmi
o3W2+yd7RSbx2qdzyc7O/Dw47hSWE0Gw9pe7aTHZIWHlhJXfnxezxfxbvTaaSZWDnXwEd90h7+t1LPIwMmPWy7FwXKM
VPbqS+gyPYa5GcnvF8g8BRjMqB5o+zrzhaPH46w8Kd7PL9JGaJW7vJQcSo2ztbh8nfMfuWX3dSDMFA3wMQ06EOCQ
O42jZg4v1jIeT3LJL+6WvHmBdLc9W/4vX4V3BDJgtovHr1Ly9/5fyFT3nXOEzqeZcCbxCIfTrlt6O1RDh08q3GWnlj
EXQkhWO4H7Lv8xq9zAhwEN5ucQEsiARVEfFOOgOpgVlW8kDC+XbiBGO+wv5gQ+q+MehlQ3UDx57Pw8cv9vNQ+c0yi
xR64Vn6fzt9LS4P/++ow8DuaJpc/hcBwZC4avLH9sN+0u+l1+J+LeO/dXRenZ9+YnE7exO+V4h4Z8FWbd6N+8U

Kcf+BwoxHAEDeCNwA4Oay/0BOSkrKMe/EiRMKDQ11WC474PzRyZMnHR7Pnz9f58+ftz/O3oYkDRkyRCNHjtS+fft
06NAhJSUlaciQIfb5KSkpWrNmjaZMmaKJEyfap6elpens2bO57kNB70geEhKS537+uc+CCg0Nlc1m0+bNm+3h4o+
yp/3www/67rvvtGjRIg0aNMg+/+DBgw7LBwcHy8PDQ8ePH7/uXgoqO5A5q8aqVat08eJFrVy50n7EXpLi4+Odsv3
8ZL9m//znP3XTTTflukxeH4ZIv38dWP/+/R2OylauXFlffffWVsrKy8gzdISEhysjI0K+//uoQcI0xOnnypFq1alW
Y3clTRkaGzpw54xBos3/vsqdZ0VNoaKiio6M1Y8aMXOdnf1gFALAGp5QDgJtr27atfH199O199c477/KdTz/wNj7Kg
uXbpIkho0aKCwsDD95z//cVju6NGj7d7ZcuW9p8kiVLZLPZ1Llz5+veZ+Fz8c8Qw/8qLN4/TZs2lfR/Hwr8eb/mz5/
C65tvvvqnMzMwi7WuXLl20d+9e7e9e9dq1y2H6kiVLZLPZ1Llz5+veZp8+fWSM0S+//OKwz9k/TZs2lfR/Hwr8eb/mz5/
v8NjX11cdO3bUf//731zvXu4M9evXV506tdfT222/ne0f3gspt34wxeuONN4q87WuJiYlRxfv3lyf/5YtW9q
PGv/ZsWPHtH379hyXJPTs2VNXrlzRokWL8qZ/Xvx59+bFSS+b/rzqg/+PdsPw1Cp3jM/ZzT90Q9ecWOV79uvLe9F1M7-/A8ISqlsatyagBabR9Cm6H4RIU2ZJhSeBAANVsHo/EQpk10UvqA3C78EBPE3PsYi4CrnCjmMwv7Tj7B0fw3/9vfBHvvwxKD

KL08Z5THTfd+myjxebwAAih///1orvxz6Z3wtmJtasmSJateurerVq+u7777ThAkTdNdddzk9bAMAAAAACofA7aZ
OnjypyZMn6+TJk6pWrZruvPNOzZgxw9VtAQAAAEDJk9dhf4uP7BO43dT48eM1fvx4V7cBAAAAACVC/qfSu0bZvGk
aAAAAAAAWK/WB243vCVdi8BwCAAAAwPVzaeDOyMjQU089paioKPn6+qp27dqaNm2asrKyirzt8uXLS5IuXbpU5G2
VddnPYfZzCgAAAAC4Npdew/3cc89p3rx5Wrx4sW644Qbt2LFDQ4YMUaMaGMGVOkbXt4eKhixYpKTk6WJPn5+cn
G97teF2OMLl26pOTkZFWsWFEeHh6ubgkAAAAA3IZLA/fWrVt12223qXfv3pKkWrVqadmyaZdqxY4dTth8WFiZJ9tC
NwqlYsaL9uQQQAAAAAFIxLA3f79u01b9487d+/X/X19d3332nr776Si+//HHKuy6elpSktLc3+ODU1VZKUnp6u9PT
0XNcJDQ1VpUqVlJGRwbXI18lmNTsnT01MeHh7KyMhwdTsoJfL7qvh05TEzj99vlHy83gAAFL+y+v9vce13XtkzNzb
jwhRqjNETTzyh5557Th4eHsrMzNSGTM0adKkXJePjY3V1KlTc0xfunSp7QTqzVy3NlUxbfunSp/Pz8rG4XAAAAAFDGXbp0SQQMHDlRKSoo
CAwPzXdalgfu9997TxdDz//PP1SV3ZcQ9dnpA9fdWsyWd/PAbQDr7PPx2dqaNm2asrKyirzt8uXLS5IuXbpU5G2
VddnPYfZzCgAAAAC4Npdew/3cc89p3rx5Wrx4sW644Qbt2LFDQ4YMUaMaGMGVOkbXt4eKhixYpKTk6WJPn5+cn
G97teF2OMLl26pOTkZFWsWFEeHh6ubgkAAAAA3IZLA/fWrVt12223qXfv3pKkWrVqadmyaZdqxY4dTth8WFiZJ9tC
NwqlYsaL9uQQQAAAAAFIxLA3f79u01b9487d+/X/X19d3332nr776Si+//HKuy6elpSktLc3+ODU1VZKUnp6u9PT
0XNcJDQ1VpUqVlJGRwbXI18lms8nT01MeHh7KyMhwdTsoJfL7qvh05TEzj99vlHy83gAAFL+y+v9vce13XtkzNzb
jwhRqjNETTzyh5557Th4eHsrMzNSGTM0adKkXJePjY3V1KlTc0xfunSp/Pz8rG4XAAAAAFDGXbp0SQMHDlRKSoo
CAwPzXdalgfu9997TxdDz//PO64YYbFB8fr7Fjx2rOnDkaNGhQjuVvzO8IdERGh06dPX3NHAZQMQUF5z0tRHjN
TUqxpBpbj9QYAoPiV1f9/i2u/U1NTFRoaWvIDIDd0EhCZOnKhRo0bZp02fFPl3vvPOO+/vrpp2uun5qaqqCgoALt
SIb97FxrlMZPLQdwWrzcAAMWvrP7/W1z7nZqaqtDQ0AIFbpde033p0iVZrY7XkZ2dnZSVleWijgAAAAAAKFouvYb7
lltu0TPPPKOaNWvqhhtu0LfffquXX35ZQ4cOdWVbAAAAAAAUGZcG7lmzZmny5Mm6//77lZCQoMjISN177716+umnX
dkWAAAAAABFxqWBOyAgQDNnztTMmTNd2QYAAAAAAE7j0mu4AQAAAAAo7QjcAAAAAAA4EYEbAAAAAAAnInADAAAAAOBEBG4AAAAAAJyIwA0AAAAAgBMRuAEAAAAAcCICNwAAAAAATkTgBgAAAADAiQjcAAAAAAA4EYEbAAAAAAAnInADAAAAAOBEBG4AAAAAAJyIwA0AAAAAgBMRuAEAAAAAcCICNwAAAAAATkTgBgAAAADAiQjcAAAAAAA4EYEbAAAAAAAnInADAAAAAOBEBG4AAAAAAJyIwA0AAAAAgBMRuAEAAAAAcCICNwAAAAAATkTgBgAAAADAiQjcAAAAAAA4EYEbAAAAAAAnInADAAAAAOBEBG4AAAAAAJyIwA0AAAAAgBMRuAEAAAAAcCICNwAAAAAATkTgBgAAAADAiQjcAAAAAAA4EYEbAAAAAAAnInADAAAAAOBEHgAAAAA4DBAAAAAAAAAAAEs4h4xl+ABAA5++fnAg2JRxUtAAAAAAAAEEqeVJCK54j3JZH1dMYCFPIM6fmBh33sZHz4iGLXr6FUx7E/XBYA/gWbvX/XbQFeW8YWXzt9eLMiZzbqRQTDPIzZIJ1BGrQJvfJ7O6XKnE=4/dFAQgPP94HHF9kqoqMl3KYDw+OvU1VABAAAAA8e1+T9YAAAAAAQfz8YkDdmmdfW2B2HIBvz7rMtQYtkGTcnA6l9dv6AmJ9YCTGhSqlPsbu7SkTU3a27ASV8M01TQfrU9kfPjeu61+p3AAAAAAAABVaR9kNe0yT6z0Fs/mFTdd7NuauNvgDvJ9uYFGyiBVaNC+VyAbF5WgUbR5MNcJD8dxoLcK/uu5cb3pjcDFzMQ+4HtLYAA5/TE3QPpNxv12kZbtvXg/eQvaj8adPjK/8UpIeoR1iq2LIB8S+/QW/AU5ZD7TLwq+PmVdtUuq9+Xy4JxA3P

AXalSJdlsthzTbTabfHx8VLduXQ0ePFhDhgwpcoMAAAAAALijQgXuyZMna8aMGerZs6dat24tY4y2b9+udevWadS
oUUpMTNRDDz2kjIwMDRs2zNk9AwAAAABQ4hUqcH/11VeaPn26HnzwQYfp8+fP1/r167VixQpFR0fr1VdfJXADAAA
AAMqkQt2l/NNPP9Wtt96aY3qXLl306aefSpJ69eqlQ4cOFa07AAAAAADcVKECd3BwsD766KMc0z/66CMFBwdLki5
evKiAgICidQcAAAAAgJsq1CnlTz/9tB566CF98cUXat26tWw2m7799lutXbtW8+bNkyRt2LBBHTt2dGqzAAAAAAC
4i0IF7mHDhqlx48b617/+pZUrV8oYo4YNGyouLk7t2rWTJD322GNObRQAAAAAAHdS6O/hjomJUUxMjDN7AQAAAAC
g1Ch04M7MzNSqVau0b98+2Ww2NW7cWP369ZOHh4cz+wMAAAAAwC0VKnAfPHhQvXr10i+//KIGDRrIGKP9+/crIiJ
CH3/8serUqePsPgEAAAAAcCuFukv56NGjVadOHR07dky7du3S7t27dfToUUVFRWn06NHO7hEAAAAAALdTqCPccXF
x2rZtm/0rwCQpJCREzz77LNd1AwAAAACgQh7h9vb21vnz53NMv3Dhgry8vIrcFAAAAAAA7q5QgbtPnz4aPny4vvn
mGxljZIzRtm3b9OCDD6pfv37O7hEAAAAAALdTqMD96quvqk6dOmrbtq18fHzk4+Ojdu3aqW7dunr55Zed3CIAAAA
AAO6nUNdwV6xYUR9++KEOHjyoffv2yRijxo0bq27dus7uDwAAAAAAt1TgwD1u3Lh852/atMn+7zlz5lx3I7NmzdI
TTzyhMWPGcJQcAAAAAOD2Chy4d+/eXaDlbDbbdTexfft2LViwQNHR0de9LgAAAAAAJVGBA/cXX3xhSQMXLlzQvff
eqzfeeEPTp0+pAYAAAAAAMWtUNdwO9OoUaPUu3dv3XrrrdcM3GlpaUpLS7M/Tk1NlSSlp6crPT1dkuTrm/f66cp
j5v/WLer61KY2talNbWpTm9rUpja1qU1t96xdUOnXsY7NGGOuu4KTvPfee5oxY4a2b98uHx8fderuBuWVWw7
Ni8uu4e7SpYu+//57h2lDhgxRw4YNNWHChGuGbQAAAAAASjKXBe6AgAA1adLEYZq/v79CQkJyTAcAAAAAwN2Uc3U
DAAAAAACURi7/WrA/2rRpk6tbAAAAADAKTjCDQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAA
AgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQA
AAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMA
NAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFi
AwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAA
AWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAA
AABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNw
AAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAU
I3AAAAAAAWIDADQAAAACABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAUI3AAAAAAAWIDADQAAAAC
ABQjcAAAAAABYgMANAAAAAIAFCNwAAAAAAFiAwA0AAAAgAVcGrhnzZqlVq1aKSAgQFWqVFH//v2VkJDgypYAAAA
AAHAKlwbuuLg4jRo1Stu2bdOGDRuUkZGhbt266eLFi65sCwAAAACAIvN0ZfF169Y5PF64cKGqVKminTt36uabb3Z
RVwAAAAAAFJ1LA/efpaSkSJKCg4NznZ+Wlqa0tDT749TUVElSenq60tPTJUm+vnlvP115zPzfukVdn9rUpja1qU1
talOb2tSmNrWp7Z61Cyr9OtaxGWPMdVewgDFGt912m3777Tdt3rw512ViY2M1derUHNOXLl0qPz8/q1sEAAAAAJR
xly5d0sCBA5WSkqLAwMB8ly0xgXvUqFH6+OOP9dVXX6lGjRq5LpPbEe6IiAidPn3avqNBQXnXSFEeM/93ZL2o61O
b2tSmNrWpTW1qU5va1KY2td2zdkGlpqYqNDTUfQL3I488olWrVunLL79UVFRUgddLTU1VUFCQw47abHkvb5THzD8
8BUVZn9rUpja1qU1talOb2tSmNrWp7Z61Cyq3HJoXl17DbYzRI488og8++ECbNm26eBEJ5tLAPWrUKC1
dulQffvihAgICdPLkSUlSUFCQfPO74h0AAAAAgBLOpd/DPXfuXKWkpKhTp06oQdVq2a/Wf58uWubAsAAAAAgCJz+Sn
lAAAAAACURi49wg0AAAAAQGlF4AYAAAAAwAIEbgAAAAALEDgBgAAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAA
AACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgA
AAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgR
uAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMA
CBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAA
AwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgA
AADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOA
GAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACx
A4AYAAAAAwAIEbgAAAAALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACBG4AAAAACxA4AYAAAAAwAIEbgAAAA
ALEDgBgAAADAAgRuAAAAAAAsQOAGAAAAAMACLg/cr7/+uqKiouTj46Mbb7xRmzdv7Mbb7xRmzdvdnVLAAAAAAAAUmUsD9/LlyV
27Fg9+eST2r17tCePXvtzp06KCePXq6NGjrwIAic2ngnjNnjv7+97/r//pnJkJzpM79atm7Zs2ZLrOmlpaUpLS7M/TklJkSSdPXtW
zKOtTm9rUpja1qU1talOb2tSmNrXds3ZBnT9/XpJkjLnmsjZTkKUscOLECVWvXl1ff/212rVrZ58+c+c+ZMLV68+Wak
JCTnWiY2N1dSpU4uzTQAAAAAAcjh27Jhq1R3tZ2uSYlm3SpEkaN26c4zc/XFWVpbOnj2yuSYlm3SpEkaN26c/XFWVpbOnj2rkJCQXNd
JTU1VRESEjh07psDAQOc2DvwBYw3FhbGG4sJYQ3FhrKG4MNbgLMYYnT9/XuHh4ddc1mWBOzQ0VB4eHjp58qTD9OT
kZFWtWjXXdby9veXt7e0wrWLFitesFRgYyC8VigVjDcWFsYbiwlhDcWGsobgw1uAMQUFBBVrOZTdN8/Ly0o033qg
NGzY4TN+wYYPDKeYAAAAALgjl55SPm7cON1///1q2bKlLkl2bKl2rZtqwULFujwULfujo0aN68MEHXdkWAAAAAABF5tLAfffdd
MmTOaNm2akpKS1KRJE61du1aygujDUUF8YaigtjDcWFsQZXcNldygEAAAAKM1
cdg03AAAAAAClGYEbAAAAAAALELgBAAAALAAgRsAAAAAAAuU2sD9+uuvKyoqSj46Ojq9vb79212rVrZ58+c+ZMLV68Wak
U3759FR4eLpvNplvNplWrVjnMN8YoNjZW4eHh8vX1vdnplWrVjnMN8YoNjZW4eHh8vX1vdnplWrVjnMN8YoNjZW4eHh8vX1VadOnfTjjz+6lm4rVmzZqlVq1YKCAhQKCAhQlSpV1L9/fUkJDgsw1i

U3759FR4eLpvNplWrVjnMN8YoNjZW4eHh8vX1VadOnfTjjz+6plm4rVmzZqlVq1YKCAhQlSpV1L9/fyUkJDgsw1i

DM8ydO1fR0dEKDAxUYGCg2rZtq08++cQ+n3EGq8yaNUs2m01jx461T2O8wRliY2Nls9kcfsLCwuzzGWcobqUycC9
fvlxjx47Vk08+qd27d6tDhw7q2bOnjh496urW4OYuXryoZs2a6V//+leu82fPnq05c+boX//6l7Zv366wsDB17dp
V58+fL+ZO4c7i4uI0atQobdu2TRs2bFBGRoa6deumixcv2pdhrMEZatSooWeffVY7Y7duzQjh07dMstt+i2226z//H
JOIMVtm/frgULFig6OtphOuMNznLDDTcoKSnJ/vP999/b4IMPOkxr2LChmThxoos6QmkkyyXz
wwQf2x1lZWSYsLMw8++yz9mlXrlwxQUFBZt68eS7oEKVFcnKykWTi4uKMMMYw1WKtSpUrmzTffZJzBEufPnzf16tU
zGzZsMB07djRjxowxxxvC+BueZMmWKadasWa7zGGGdwhVJ3hPvq1avauXXOnunXr5Xr5jC9W7du2rJli4u6QlmQmQJiokyd
POow9b29vdezYkbGHIklJSZEkBQcHS2SZKswRqZmZl67733PHiRbVt25ZxBkuMGjVVRqZWmPiojRs3tv/xyTiDs7z33nHPMYanKFBgwa
Kj4/XuXPntGLFCg0aNEhxcXH2+YwzOMOxY8c0ZswrV+/Xj4+Pnkux3hDUfXs2dP+76ZNm6pt27aqU6eOFi9erJt
uukkS4wzFq9SdUh4aGioPD48cR5s7Wzza1uYX9N8uZz55RKtXr9YXXl3eqlu3rlq
2bKlZs2apWbNmeuWVVxhnc0Kqd3cqOTlZN954ozw9ozw9PeXp6am4uDi9+uqr8vT0t6s5NzfJoFOFP2HTAZe3CWRx55RKtXr9YXX3yhGjVq2Kcz1uBMXl5eqly
2bKlZs2apWbNmeuWVVxhncKqdO3cqOTlZN954ozw9ozw9PeXp6am4uDi9+uqr8vT0t6s5NzfJoFOFP2HTAZe3CWRx55RKtXr9YXX3yhGjVq2Kcz1uBMXl5eqlu3rlq
LlLrA7eXlpRtvvFEbNmxwmL5hwwa1a9fORV2hLIiKilJYWJjD2Lt69ari4uIYe7hg9/PDDWrlypTZu3KioqC
H+Yw1WMkYo7S0NMYZnKpLly76/vvvFR8fb/9p2bKlBg4cqPj4eNWuXZvxBqdIS0vT3r17Va1aNY5ncKr27dtrCVhcg0EiQ5qWJ0nXnuc2Z5y3vUvVp2+yHUAsQMBPZGU72YCWzYVuh2Rw0AcpT6e67I2y1gUFS6Ls7HRhmi60eqSRjfPdfsXZUtdDFRbHHY5tXhJp9c
bp/vvv18tW7ZU27ZttWDBAh09eRbsNBLr2sUVu0yxxJtXjKxhtatD9N1/17XkdUHsaePcyF2Rr
6mjlzpvz8/DRw4EAXdg13M2rUKC1ulQffvihAgIC7J/EBwUFydfX1/7dtYw1FNUTTzyhnj17Ki1QufPn9d7772
nTZs2ad26ddg3dEVAQID9PhTZ/P39FRISYr/L9KH8MAMLxLG8cXQuqU7F8z9I2h4y5xhQPObw2ot7oq5Hl3WIzqg5c4ewsDCmOTz87
7bp05c0bTpk1TUlKSmjRporVr16pmzZrvLtDUOJ0qYkZ9xOEOKlpzpTgrPMAnMFOCYECxtwl1PXoOIL0bA9fvB5S9omNGVOMbG6MxMCOlDGk7zEXcakJEt
mTRutX79eAQEBrmoZbmju3LmSpE6dOjlMX7hwoQYPHixJjDMUWVpamh577DGdPHlSfn5+atasmVavXq1OnTpJcAgn4wP5fcn/v/8/+P/qMw8OOQYIEKREdCvemQU9s9b8R0NF
UL8YbnOH48eMaMGCAT/ODg+rcqVK+umm27Stm3bcA6YrRV4PQiHLBUOfY0cNJqLmvPKbBeXhcNkVdyPkCYvu4xPFUikE3u0YhxFCsyz2bi/lIAfmrlcEoPdDMEyhKWP9d27
BAjcAAAAAABYgcAMAAAAAYAECNwAAAAAAFiBwAwAAAABgAQI3AAAAAAAWIHADAAAAAGABAjcAoMw4fPiwbbDab4uP
jXd1KgXTq1eljx451dRsAAKCQCNwAAJQigwcPVsAAAOfIysrSxx9Dv5n9PVDQAAAOfIysrSxx9/rNWrV7uFQA
AII5wAwBKoaysLD333HOqW7cuvvL/Pzz7eBcB/PUdvXxpJjD15kvvVbiblzf/ZUSnssCjhZRs8u6s2Yb4qvG5MlTvqtU5BFwdXm1gQdi0JEhU2fsVeqeJbmbt4cIE/PjCUEXnNUV0l+P6Yd8dcVakN4N2ZyZUdUnVGncAdz2O2HOX6P8WZ6+xxGGmbtpCQJFYQ
1bdpUy5Ytc9h+p0+p06dNHr0i91k0gPW17lfr7/+uurVqycpVd0bxwxc1tKDx2C0gY8KmkR/M1pCvUg/G6
rVtX999+v06dP57lfr7/+uurVqycpVd9xxh8P8r7/+WuXKtmpkScePH9c9e5+Pc999yj4OBgVahQIe9+m2xr
bG6vmzZtr/vz5ioIkJ+fn68deadX9aOoE6dNCtW5Up06d5O/vr8qVK6us2++P5w+gSTyB79aO
5+voqIiJCo0eP1pP1sWLF+3Za9WqpZ2Z2ro0KEKEAhQzZo1NXHPKIKVLFi19ks9nUqVMnTVrcvPZvMVHvxerzZ
7q2LFioqJidGRI0Tl48Fd4VzJXSbmIB2xHVyDtdGlAwHdDaeBQ8bAO/b/lpRJUtj8wwdAG0Tb3Qb7JywhNxp6ezgNLBNP+wfqVNCtBnGLi+RsvvjH1tcZiMbSIzB1
33GEiIyNNenq6McaY48ePm+eff97s3r3b/P/Pzzm+Fz3U0QaHDZj6tl3Qj/N5gN2bt1ddD8+ZVi0Qh/tXrIz+eNPzcemzgLsOJ+6JwS2Wfqrr3j88fe0ZFCxN
69evty0gyNWrUMEuXLjUHDhwwo0ePNhuqVDBnzpwxahz4sQJExoaiZNmmT27dtndu3aZbp27Wo6d+7sUGfMdDH
GGGO2b99uPDw8zNKlS83hw4fNrl27TS83QPUUQ8FPrr12w3hm5YCzgf4P/rJ1+9/bwx5vz586+ZQ7dqZmWbAP6gZPMFiLzN2B1V/H9nZu4Ew2aZ/7SzujZY9Hr7+/LLeuI
WLcYYY1auXGnXly9vXnvtNZQkGBefPC+F+hJYTZu03OiwDYVqzMlLy8Hz7MLVRuHDjgYjAQRdV3K2N4zCQcunkpbnzDQrqTTHyyuzUnZt3+4
1N910k+nRo4d9nSSlTphh/f39zymJ3zzYQ23N2tdrs5mLZmZN2cDgGMpAjwo6bp4+NMnZqM2JTbtyMGl1Yc9GrGqZs2uUMYTgpsNhqaLDvAMKLWTzJMM5A/f
vN8aYAr1Oe/bsMRUqPDAwfvSS2b9v/n6669NixBgNBsWdD1Ce/n+9Giqc0MRtyTBNSlsHIns39ZzGzMDVHsl9HIzYMV4HkG1oFdMvwHoj2t6Xb02mHAvcxDhHscnXpHzmj7Jy
GfPvtt0aS+eyzz0uaMSU9PN0FBQebxxx8W0S0eNHv37CQjRsGt3AQ0CamBzVsnyxK5nwS7zWyfDzTOPXJBUJxD80TEg4xu3Hvqjx9WEckTSm4kwoWv7l3VJ3HUQaaAqrxxvnfYnv
2b7zxRo552UHuzTfftE/78jfjcSR7CAvtNNr169zGOPPzVZ3BzDQgF4xJJ8QTKA7LxHyyLNFWmgBjBSbUydJqPOhJYoUc1lXqHVNHrUxTUbdn7UKwXUkJfUFDrF94UM8
bzXzyySfGGGOefvpl061bN/vc+7pszF6v0gSsTdE81Jq40r2Za6s65BtHwMQCSkQHcr0kOkuSdhUNMDUMZbmPZGtKc/tQ08U8cN1yMyE5mbNDvaj9EBC16/gVS68K2EK5eY
ftWvXzgwbNsh2p133ml69eqV5z3s3brVJvfWfdqyZcuJMj4+PfGUKVOM4K9IH8aVuEq2Vm3XeffX5WVpapUqWKmTtJ3Dx4dlnV2fVqcfG1/C22GmDMeSgyq5dbd4wYqkKMr
MMcaEh4ebGTNmONRu1aqVGTlypDGmDYK/T/fffb4YPH+6wqeYH/4z5tyiqZ33wCG2N+D9z33XefffX5WVbpapUqWKmTtydTHyM/8nTQKgHpdDSTBqK2eSkpK
3rkOd3bt325c5c+aMkWQ2b+2dxUcY/MGAmd8GEXqKydbGrsWzfc8nUiRe/7F9dLA0UPAc4CZeJR49xbyvkqK48R0yqrL0PGzm/L2t4XmOFN5sXKS7CtP3Lly55Lll7w3qUvC7
D0dHRCgkJUYKFbR+/XodPXo21kbyd7G7kt4+/vr4CAAPsyfrtrcUfb3/GznaZd4UOs3vXybEI1IxP1u1CdfD99hvmWgvPXZv7OtcalcoK6PqSbxnOwsQSRUpOfq/4r58hmUYyM
Vu3Zt3X///Xr33Xd16dIlh0+3vIlBxi9VPlS8RooemT3D9CZ/sRi9VYn7GDyMtPUIR2XTyGG9FxyRrIYXZ7vF8hoUAvJX+KLVbeMuzlVkA6ZPS89SSlxf+Dt6ybDf
duXJFqamp9mk1a9ZRjRo17I/I/btzS5I//btm2rrKwsJSQKDU1VSdOnLZqNHz2n9+nXbu2K+XzcbGDwUwD5w6JDaMYVlaWEhMTz9WDQ9W1ZZUW2crXh
TWFhYjtfpj4KDgzV48B48l95dffvV21SuvvKKQK+k2dwuVlaJgAQeLr6GnM1u7wwNCu3a2/t81+i/2Bds/D8cWS4OKLL+q
ll17S+PHjHtxHjRsXHx6t79+Kh68NmvH02MTKMmN8uZ3X2kmm2/O0XNAQIb27dqlzcuWqVqQuW/qY8iLuLz3bNq39BQfnbJ/KsQbmTaT/W5/3Ai0DrQcO8Ohi4Z1PCDb0nKPsqyL2HWAdH6tQnnbx
1apo8ebKaNWtmvzcZq69jq4q1q9bHQ12ajfb3mpMo/mL6S8Ob6QNKDHNmBeeKHQ9fk+lTfQ0YqZ1z12q3vmuQ33u6zSkKoLyQy55SJUz9WrVbDmx64/Tcno0
1ct1G9nby61WSFi5crQxK1bt6pdu3ZavnsTX9W26asKBTI4nTDgcStttrZAKNlt6S5X46GunLvpPjeRT1hgc2HLmmyVTjxR2LN5SzXqFEf1LP0WnwNLv0VnMw+iB
3331q1qyZat7eurQMHDji5S+kvf/mLzvxR9WqVUt169Z1Z1+ZH39891HU9PT916662aPXu29uzZo8Zo8OHD2vjxopSpA8
//FD9+vWzLxsdHa34+HidHidHHdxPXsZTqLNkFXWB6TsuhSJpP3FXtmf3iAI3KQzSkxGOj+xRGmFb9m8OI3fjvtnHeo2OqpNFJsJsuAUtsOfb12O4BWc8xsB5PfeyzcFM+J4qaqLzyuY
8vMi1s5+/Pz93devW16XbZ1ZmEoy15dFBDhkR2pFmZ3yYKbmH/kWJLlmsqGhPWN2sqvMSFoUffLAy0K1kNYYl4GHjcxSpBe7xZvvTPNkwbBgHCtxdBGYRRNGgexQQ1y
P15IlS/Tzzz9pb6srZz7cm6tkvGs1MK+LZI+rXWLvd98SRvsvOT/ItZ3WZbG3q9JSzCzMcrGxXtxyBWKSlGzG3iMfuWs+oBRYJvpvCX7VqVu0RwxNWkSgK0xm8hj8/p3LyfSWFpLvl+eBtwxk6X2J+n8gj5Og6XkvNq3nQt
1Gjp0aK4BcM2aNXr11bcVHx+vI0eOaMmSJcrMKyocj0d5+Pr1+TSFrhm7ICvmi+0YpvKybxNT6jpRwp+PaAo0lwZXvDd85lGzsyHNq7HZ+vjR+7mk1nRBBuZQL4DY0iQ8hRMjb/7XHZPChzqR0HmqmcS3aP4

rVtjv9P2Pf/xDixYt0rx583TgwAHNmTNHK1eu1OOPP17kffPx8dGgQYP03XffafPmzRo9erTuuusuhYWF2Ws/99x
zWr58uRISEjRx4kTFx8drzJgxBa4xYcIEbd26VaNGjbKfGbB69Wo98sgjBd5GlSpV5Ovrq3Xr1unUqVNKSUlRYmK
iJk2apK1bt+rIkSNav3699u/f75QPIgAAZQ/fww0AKHWefvppeXp6avLkyTpx4oSqVaumqnv37vL
z89Pw4cPVv39/paSkOLXH8PBwff3115owYYK6d++utLQ0RUZGqkePHipXLufn4XLufn4RUrVtTKlSsVGxurK1euqF69elq
2bJluuOEGvfXWW2rTpo2qVKliX97Ly0vr16/XY489pl69eikjI0ONGzfWa6+9Jknq37+XnnlFT3//PMaPXq0oqK
itHDhQvtXYxVF3bp19de//lW9evXS2bNn1atX9de//lW9evXS2bNn1atX4BrR0dG
Ki4vTk08+qQ4dOsgYozp16uuu8u8u8DY8PT316quvavvfHvmI6dOi7cuX66eftLixYt15swZVatWTQ8//LB
GjBhxXc8BAACSZDPGGFc3AQAACq9fv35q3769xo8+pWFBsbq1WrVik+Pt7VrQAA4HKcUg4AgJtr3769BgwY4Oo
2AADn3BKOQAAbq4kHNkGAAA5cUo5AAAAAAAW4JRyAAAAAAAsQOAGAAAAAMACBG4AAAAAAACxA4AYAAAAAwAIEbgA
AAAAALEDgBgAAAADAAgRuAAAAAAAAsQOAGAAAAAMAC/x+kNi0cFMACRQAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 1200x500 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"# Plot the log-vars\n",
"plot_logvar(trials_logvar)"
]
},
{
"cell_type": "code",
"execution_count": 27,
"metadata": {},
"outputs": [],
"source": [
"def plot_scatter(left, right):\n",
"   plt.figure()\n",
"   plt.scatter(left[0,:], left[-1,:], color='b')\n",
"   plt.scatter(right[0,:], right[-1,:], color='r')\n",
"   plt.xlabel('Last channel')\n",
"   plt.ylabel('First channel')\n",
"   plt.legend(cl_lab)"
]
},
{
"cell_type": "code",
"execution_count": 28,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkAAAAGwCAYAAABB4NqyAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABn+UlEQVR
4nO3de3gTZdo/8O80PXEsUBAKDRRQiyAI76ICUoEfqKi4SC0grFg8rq/u2qorwouviifA1wOs666ry4IrtiqkiKu
rHLTlLC5IkV1ZRChQShF1ofWwE2e3x8xoUkmyUwykzzk+7muXNDJZDKHpDP3ee7nfiQhhAAAREREkxegeIiIi
IEo0BEBEREBESUdBkBEBESUdBgAEREREUdJhAEREREREJhwEQERERJR0GQEREREJR0UUo3eATPyeDw4cuQI2rRpA0mSjN4
dIiIiUkAIgW+//RZdu3ZFSkrrkNh4GQDKOHDkCp9Np9G4QERFRDGpra5GbmxtxHQZA2NtbxtxHQZAMtq0aQPAewLbtm1r8N4QERG
REo2NjXA6nf77eCQMgGT4ur3atm3LAIiIiKiFUZK+wCRoIiIiSiwMAihaiIikg4DICIiko6DICIiIgo6TAHiIiIKIE8Hg9Onz5
t9G5YVnp6etQh7koaowACIiqQ06dPo6amhBh6Px+hdsayUlBT07NkT6enpcbXDAEhERAREVECCCFQX18Ph8MB+Ph8MBp9OpSSt

GsvEVKq6vr0f37t3jKlbMAIiIiCgBmpqa8MMPP6Br165o2bKl0btjWZ06dcKRI0fQ1NSEtLS0mLfD8JOIiCgB3G4
3AMTddZPsfOfPdz5jxQCIiIgogTjHZHy0On/sAiMiMiu3G9iwAaivB3JygIICwOEweq+IbIEBEBGRGVVUACUlwOH
DZ5bl5gILFwKFhcbtF5FNsAuMiMhsKiqAoqLA4AcA6uq8yysqjNkvSkojR45EaWmp4vXfeustnH322XA4HKpel2g
MgIiIzMTt9rb8CBH6nG9Zaal3PUpKbjdQVQWUl3v/NdtH4Ze//CWKiopQW1uLxx57DNOnT8e1115r9G6FYABERGQ
mGzaEtvw0JwRQW+tdj5JORQWQlweMGgVMner9Ny/PPI2C3333HY4dO4YrrrgCXbt2RZs2bYzepbAYABERmUl9vbb
rkW2YoWf09OnTmDFjBrp164ZWrVrh4osvRlVVFQCgqqrKH/D8v//3/yBJEkaOHIlXXnkFK1euhCRJkCTJv77RmAR
NRGQmOTnarke2EK1nVJK8PaPjx+s7UPCmm27CgQMH8Prrr6Nr165YsWIFxo4di127dmHYsGs2cP8vPz4XK5MGz
YMLRs2RK33XYbGhsbsXjxYgBAhw4d9NtBFdgCRERkJgUF3tFe4WqdSBLgdHrXo6Rhhp7Rffv2oby8HMuWLUNBQQF
69+6N3/zmNxg+fDgWL16M9PR0nHXWWQC8QU6XLl3Qtm1btGjRAhkZGejSpQu6dOlimkKQbAEiIjITh8M71L2oyBv
sNP+T3xcULVjAekBJxgw9o5988gmEEDj33dn6vbFOGAAREZlNYSGwfLl8HaAFC1gHKAmZoWfU4/H
A4XBg+/btcAQF4K1bt9bvjXXCAIiIyIwKC70JHawETTjM1pXJ58HJEne5/XsGR00aBDcbjeOHTuGAhVvlJ6eHve
8XXpgAEREZFYOBzBypNF7QSZghp7Rc889F7/4xS9w44034plnnsGgQYPw9ddf48MPP0T//v1x1VVXyb4uLy8Pq1a
twp49e5CdnY2sKy4ZnHXCpOgiYiILMDXM9qtW+Dy3Fzv8kT0jC5evBg33ngj7rvvPuTn5+PnP/85tm7dCqfTGfY
1t912G/Lz8zF48GB06tQJmzZt0n9HFZCEkGtMS26NjY3IyspCQ0MD2rZta/TuEBGRDZw8eRI1NTXo2bMn5MjMzY95
Oss+RG+k8qrl/swuMiIjIQtgzqg12gREREVHSYQBERERESYcBEBERESUdBkBERESUdAwNgL799luaiR48eaNG
iBYYNG4a///3vEV+zbt06uVz37IiMyIEMDoFtvvRV1r7dHfsdVLs1eC2eMA6AhMvH/rdLz+3rfJeCvpVScYHX3
zJ6ZNm4ZJkyZh69atiTosIiMjnD6gD95z//QZs2bbBy5UpcffXV/uUDDw7EuHHsDbb7+N3bt
3+5fdcccd2LlzJ7Zs2QIAmDx5MhobsG/Hee+/51xk7dizat2+PRo0a2D8A5guu+++9CvXz/
U19ejvr4ekydP1u14AI07wFatWgUhBPLz8/HFF1/g/vvvR35+Pm666SYA3tycuro6/OUvfwHgHfH1u9/9Dvfeey9
uu+02bNmyBYYsWLQoY3VVSUoJLL70U8+fPx/jx47Fy5UqsXbs2bGI1ERGR6bndQEkJIDdwWWhAkoDSUmD8eF26
++2w89dRTss+99tprkCQJL7/8Mjl3zfC2fhtO7/1EebRgmbCBwqZzUlJIYLHDKZkDmXDfAsEI10HqCkmVpm3EMS3WFpRP+sWDHHi8+mG5I7Bm1yeDd2NGBeIpeqYrQpR8BBNBGT47Ah9Cuvh7sGHG5yL1KU
TUxPWvWZoC1BDQwPuuusu9OnTBzfeeCOGDx+O1atX+1tz6uvrcejQIf/6PXv2xKpv7vx6BQCBEOYeVyr1VvSNT/9ycXqVz3A3+AQ7aW09Dc2JtSkqT8d0pKxOcHP4KDUdlk/jz8j4hp3XbsMzLkgRjoe5Tw+BkqqJ5Kw
W1113nX+dYcOG4fXXX8fixYsxYMAALFmyBG+88QYuvvjihB8fERHZgNsNVFUB5eXef91uTW8QRRv1AB4tpYGjPqTF8D4rkPUE5zEOXXCazvDI9zb/BOMNdgRxkWtbZV19LkXsrg/94sN8jVa80XFCvXOAx92DeaFDIThGHm5YQ1r1RESdCZohmJXJN/TVMwbQm9GyDoDW3s8EJpOHMxUIUUa1mNnHwp1G4O7kvYIBtbbmbdcktJNSfpHxSTjcC6pVGnqV2I+b2Yqy7lB7tjxu8mwAmnB2JJNLgKSM5h4e7oRDiFCy7FMM4hs9WfYyv09yMC9kl5d8yQq3P0NDhj0PYXWfpgkYSL6sD40Gc+1RBB+7rnw4Pgxa2nlINK09TgYhkXUwnBLtSkk+iMIGmQO66WkAlHs/I/SnwYVKRhlKvTLxtGEMEOVSrC72pJ0K1+2pvj7aTKFTB9w9xV8Vrm/yfiKIB5JOaVHw+6NQdx9fBmj8hNiE/ACL4bF2KsgSnRhcX0mbR3IOF7/Zkz3HF/NAeBmrgaH5hhSOP38N4XUXzHLvlHbdh2MAK/BhKvbYVQyyfCFNUiD5NQPQ/zjZsoF9JuGFL7i4+rtKUDOexHSVtUyhONOKvumHZzh3rhNOdyH+y0zM+U2dXg4nhCUMk3UVOLxKK+f5MGazi7UmOVQtzaqxPbPwVAFLkxAOmx4uNzwVonrpBHjYSyyuYzYnnzJJQTOUWyeI3cIO4VCDhk1ErEYR/PD8xQt7lhKhw1gjsyRYLc+FU4xYYCBpwQrCPmEfjgsZsF+KSvVGQOKbxtEojsJvnHJSbYkJ9xKyHwsN0Lh9MgGRK1v/94eSRFDf56CPW/H99

PomfPnmjRogV69eqFRx99FB6PJ+xrpk+fDkmSQh79+vXzr7NkyRLZdU6ePJmIwyIiA/kn9MyowMjpeXCMYVloq/E
VtQwubVBX513OS0haMHQusPnz5+PFF1/EK6+8gn79+mHbtm246aabkJWVhZKSEtnXLFy4EPPmzfP/3NTUhAsuuAA
TJ04MWK9t27bYs2dPwLLMzEztD4KIzCdcWWjfHZR9YqbldntbfuT6JoTw5nSVlgLjx7Mxj+JjaAC0ZcsWjB8/Hld
ffTUAIC8vD+Xl5di2bVvY12RlZSErK8v/81tvvYXjx4/jpptuClhPkiR06dJF0X6cOnUKp06d8v/c2Nio5jCIyEx
4B7U0NUUtk3B+WNKQoV1gw4cPxwcffIDPP/8cALBz505s3LgRV111leJtLFq0CGPGjEGPHj0Cln/33Xfo0aMHcnN
zMW7cOOzYsSPsNubOnesPrLKysuB0OmM7ICIyHstCWxqLWlKiGBoAPfDAA5gyZQr69OmDtLQ0DBo0CKWlpZgyZYq
i19fX1+O9997DrbfeGrC8T58+WLJkCd5++22Ul5cjMzMTl1xyCfbu3Su7nVmzZqGhocH/qK2tjfvYiMggvINaGot
aUqIY2gX2xhtvYOnSpSgrK0O/fv1QXV2N0tJSdO3aFcXFxVFfv2TJErRr1w7XXnttwPIhQ4ZgyJAh/p8vueQS/Nd
//Reef/55/Pa3vw3ZTkZGBjIyMuI+HiIyAbvfQW0+LbqvqGVdnXwvpm9iWxa1pHgZGgDdf//9mDlzJq6//noAQP/
+/XHw4EHMnTs3agAkhMCf//xnTJs2Denp6RHXTUlJwYUXXhi2BYiIbMTOd9AkGBvuK2pZVOS9VM0vIYtakpYM7QL
74YcfkJISuAsOhyPiMHifdevW4YsvvsAtt9wSdV0hBKqrq5Fj1b/4iEg5u5aFTqKx4SxqSYlgaAB0zTXX4IknnsC
7776LAwcOYMWKFXj22WcxqLWlKiGBoAPfDAA5gyZQr69OmDtLQ0DBo0CKWlpZgyZYqNO+64Q9fjISKTsNsdNNrINsA7ss1GpZILC4EDB4DKSqCszPtvTY31Lh2Zl6FdYM8//zz+93//F3feeSeOHTuGrl2
74pe//CUeeugh/zr19fU4dOhQwOsaGhrgcrmw0PdXXpATJ07g9ttvx9GjR5GVlYVBgwZh/fr1uOiii3Q9HiIykcJ
C71B3O+TLJOnYcF9RSyI9cCoMGZwKg4hMpbzcW806mrIyQOEoWiI7ssxUGEREpIDdR7YRGYABEBGR2flGtgUndft
IEuB0WnNkG5FBGAAREZmdXUe2ERmIARARkRXYbWQbkcEMHQVGREQq2GlkG5HBGAARUXKy6pQSHBtOpAkGQESUfJJ
gSgkiiow5QESUXJJoSgkiCo8BEBHZk9sNVFV5iwhWVXl/TsIpJYhIHrvAiMh+wnVx3Xab7aeUsGpqUyR2PCYyHgM
gIrIXXxdXcCtPXR3w8MPKtlFfr/1+JYAdU5vseExkDuwCIyL15LqXzEBJF5cSFpxSwvSpTTF8Zkx/TGRpnAxVBid
DJYrAzH+SV1UBo0bF/npJ8h5LTY2l+ljcbiAvL3zvnuGHFcNnxvTHRKbEyVCJSB9m/5NcTdeVjaaU2LBBeWpTwsX
4mTH1MQUza4soRcQAiIiUscIIKqVdV3Pm2GpKCaVxX8JTm+L4zJj2mIJVVHibqkaNAqZO9f6bl2f8HwMUFQMgIlL
GCn+SK501ffZs4MABoLISKCvz/ltTY8ngB1Ae9yU8tSmOz4xpj6k5s7eIUkQMgIhIGSv8Sa5m1nTflBJTpnj/tVi
3V3NK476CgsTuVzyfGdMek48VWkQpIgZARKSMJf4kR1LOmq4m7kuoOD4zpj0mHyu0iFJEDICISBnT/0neTGGhrbq
4lDBl3BfnZ8aUx+RjhRZRioiFEIlIGd+f5EVF3htX86Z/U/xJHiQJZ00vLATGjzdR1WQNPjOmOyYfq7SIUlisAyS
DdYCIIpCr6eJ0em9kNm5hoTjY8TPjK1RUVyefB8RCRYZQc/9mACSDARBRFJycKXnFeu3t+JnxjQID5Fu3DO+nSz4
MgOLEAIjIHOx4z7Q0M1cBN4odW7csjAFQnBgAERmP91qTCTfJLFs7GKmbCAOgODEAIjIW77Umw4m5yCI4FxgRWZb
Z68sl5bRPrHlDNsQAiIhMxcz32qSd9ok1b8iGWAeIyGasno5ginutzEmsWOmQ7ZbzTfukSbecWS8ea96QDTEAIrI
ROyQOG36vlTmJIjcX7/1nIYQIPYlCeFNgSku9BftijlfMfPF8FZ2j1bwxQxVwIoXYBUZkE3aZmNrQGTfCncTDdfj
jN0WYAPmTGHe3nNkvnukn5iJSjwEQkQ2YPXFYDcPutRFOogTvsgUoRQrCn8SYuuWscvFMPTEXkXoMgIhswMyJw7E
w5F4b5SSmQKA7alGA8Ccxpm45K128JJxkVomkHBloA8wBIrIBUyQOayzhk2AqPDk5CF0vrhQYq128JJxkNhIzp25
RZAyAiGzA8MRhnST0Xqvw5BxF4Hpxd8vZ9eLFwqyj4MIIV7BT05GBpBtWgpbBStBkNZyYWgMKTuIPHXJxXmYNDtc
BBdiAHNTD3SkH179QgMKJMZ5YXjwvizWlsDi2ObESNFGS4SAdDSg4iS1fWoD9C1bi+055qMIolGMq3vxqFArvzYt
9pBYvnvlHwcmwUuoWyWMARGQTHKSjQris1WgnEYBjUhEyv9L4Rp3MF88qo+CCWC11i0KxC0wGu8DIyiyWRpF4Srp
a5E4ioH+fRzJevKoq75wi0VRWmir52qK7bXtq7t9MgiayGQ7SiUBp1qrcSayqUt7nEesFSMaLZ9GmFBbHtj5Du8C
amprw4IMPomfPnmjRogV69eqFRx99FB6PJ+xrqqqqIElSyONf//pXwHoulwt9+/ZFRkYG+vbtixVuh9OERkZlG
6WoQARElp+K4Wi96oTc+io+CYumV9hgZA8+fPx4svvojf/e532L17N5566in9H55//vmor92zZw/q6+v9j3P
OOcf/3JYtWzB58mRMmzYNO3fuxLRp0zBp0iRs3bpWiHoulwt9+/ZFRkYG+vbtixVuh9OERkZlG6WoQARElp+K4Wi96oTc+io+CYumV9hgZA8+fPx4svvojf/e532L17N5566in9H55//vmor92zZw/q6+v9j3P
OOcf/3JYtWzB58mRMmzYNO3fuxLRp0zBp0iRs3bpWiHoulwt9+/ZFRkYG+vbtixVuh9OERkZlG6WoQARElp+K4Wi96oTc+io+CYumV9hgZA8+fPx4svvojf/e532L17N5566in9H55//vmor92zZw/q6+v9j3P
WHRiaAzRu3Dh07twZixYt8i+77rrr0LJlS7z66quyr6mqqsKoUaNw/PhxtGvXTnadyZMno7xGxEe+9955/2dixY9G
+fXuUl5dH3S/mAJGpWGx4sGmVlwNTp0ZdbSrKUOSaEnpqOVxdP76uSSDw3PqCIpNHE8mYumVWlhkGP3z4cHzwwQf
4/PPPAQA7d+7Exo0bcHBwgCIpNHE8mYumVWlhkGP3z4cHzwwQf
gweHBpqWwZaYeOfKDjtjnoR+LN6X4UremTPH+y4+ANRgaAD3wwAOYMmUK+vTpg7S0NAwaNAawaNAilpaWYMmVK2Nfk5OT
gpZdegsvlQkVFBfLz8zF69GisX7/ev87Ro0fRuXPngeld3m3LlzkZWV5X84nU5tDpAoHhYdHmxaP3W
1CMh3tXgg4RCcWI+C8PVbLH6jNjXOM0YJZugosDfeeANLly5FWVkZ+vXrh+rqapSWlqJr164oLi6WfU1+fj7P
9Pw8dOhS1tbV4+umncml/qXS0F/oQkhQpb5zJo1C/fee6//58bGRgZBZDw1ldaSbeRQLHwtONcVwQMJKTgTWHp
+CopKsQAeeP98D5vLnPBJypJIMo6CI8MYGgDdf//9mDlzJq6//noAQP/+/XHw4EHMnTs3AkZ8iQIVi6dKn/5y5
duoS09hw7diykVcgnIyMDGRkZMRwBkY6SddSRngkVhYX455zlyHq4BE6cCS4PIxelWIAVONPaELHHLAlu1MxrIbs
zNAD64YcfkJIS2AvncDgiDoOXs2PHDuQ0+201dOhQrFmzBvfcc1H2erVqzFs2LD4dpjszWy/8U0w6ijhpyQBCd/
nzS5Er5fGo1fdBnRBPeqRgw0o8Lf8sH4L8+4pSQgDFRcXi27duol33nlHH1TUiIqKCtGxY0cxY8aMzM/zozZ84U06Z
N8//83HPPiRUrVojPP/9c9AIL5fKvs2nTJuFwOMS8efPE7PE7t27xbx580RQJZ5WRVnrcqSPUGUqBgAcSFqAUKzHenyDcnKanRMOEb96XY6f0MvTfWwE8omBSWSITU9QFlpKSAkmSEG5V33O

SJMFtg6Js7AJLIgZ1vZiCwmMfiUqsw8iQ5ZqeEo2mmQg32btFZlQwnJLL0L2bGzXIg2TmbmNKWpp3gdXU1GiyY0S
m45uIMdqN145johU24eQgcD1dTomvSGFRkfcN5OaDijLNRLTC2ZLkLZw9fjzvy+EouQx/uW0DpIdN3m1MpICiAKh
Hjx567weRMTS48VqWirmxfHQ9Jb5pJoL7r7p1U9R/FXM6V4SEoWTM8Q13GXJzvdf90lMW6TbWQTJ+HuwsprnAXn3
1VVxyySXo2rUrDh48CABYsGABVq5cqenOESVEss7v5Gv9CjNFjADwVUonbMKZAqIJOSXBTTgKB6rGlM5VUeHt8xk
1yjtT/KhR3p8rKiI9ZXsRp+UyQYFOIyTz58G21I6x//3vfy86duwoHn/8cdGiRQuxb98+IYQQixcvFiNHjlS7OVN
iHaAkFVQLJymKnYSrM9Ts8Z9OuWJ9qUv/UxJnPabKyvD1a5o/Kiujv58HkiiEy5SloQxncIFOI5i5VBgF0rUQ4nn
nnSdWrFghhBCidevW/gBo165dIjs7W+3mTIkBECUVuYqAif4tH60CHyBEdrYQa9eGvbGqui9HeT83JHEQTpGCpmS
4v6uXRKW0WRzSWtTcv1V3gdXU1GDQoEEhyzMyMvD999/H3SJFRAlWWAjs2wd06iT/vK8LqrTUmwShh2gJPADwzTf
AmDFh+x186VxAaK9eSO5SlPdLgUB31KIAG0Kea55LlLTM0G3sdntHMpaXe//V6bOpJreMrEV1ANSzZ09UV1eHLH/
vvffQt29fLfaJiBJt82bgq6/CP6/3b3k1CbO+WjMyQZDi+3KMI+Cas2GOrzoRE4U0FhzsLFuWsIQcq5QKI/VUV4K
+//77cdddd+HkyZMQQuDjjz9GeXk55s6diz/96U967CORrjiyA8b/lleTMBtlTLuiwtkxjICLZ5ftIvS74oBD76H
ucpUt5ehUhDFJc76TQyx9bC+99JLo3r27kCRJSJIkcnNzxZ/+9KdYNmVKzAFKHmacBN4QqjOIYxM2zzxaAo/W+xP
l/ZIuB0jBAABDvivhso8TmJCThDnflpaw2eC/+uor8eWXX8azCVNiAJQcOLKjmQT8lo96A1UwIi3kUVYW+zFHSOT
1jQIzNMc3UaMSFUQ2hnxXlCTG6xSoB0uinG/LS1gAZFcMgOyPIztkKPktr/Km7Fu9tDTy/SogCFJz04v3Rif3fk6
nEC5XpKf0l6jmlmgtLHPmiKZTTcZ8V5S2SmodGIdh6OeBFNM1ADp69Ki44YYbRE5OjnA4HCIlJSXgYQcMgOwvQT0
+1hPpt7zKm7KaWCY7O6g7bO1aITp0SMxdN0JQZ0hpqEQ1tyhsYflPx25igkxNJN2/K2VlsQdAOn1xk7FUmNWouX+
rToKePn06Dh06hP/93/9FTk4OpDBVZInMzOicX9MKl0G8cqX8LKPBiac/Zcl+vLIezy/IwREUADIzyQf75hvv4J7
Ro+HNVh49Gnj5Ze+2gcD31Xo+Docj7JxVEZ7SRyInNFNSegBAxtd1WI4iFGE5ViB8crHm35VYsop1nrsv4Z8H0pX
qAGjjxo3YsGEDBg4cqMPuECUGR3ZEEPxbXulN2eMB7rkHOHwYFwGoBFCLXJRgYcQbp48/APKJNimVHacoiXlCsxg
ojFgkAALAApRiJcbDEyag1fy7Em2i4mB2n7uPNKe6DpDT6YRQ8mEkMrEo02BBkgCn056TwKum9KY8cWLIet3gbT2
YgBjrs+hVayZBRfRUS2TTpIqIJVJhSN2+K5EqW8qx+9x9pDnVAdCCBQswc+ZMHDhwQIfdIUoMVVWDrUjLG3wcN9s
UeP9YWoBSpCDyPoRt0PC1SE2Z4v033oti5lktE9k0Ge2vABldgwpD6v5dCVfZ0un0FkNMRBFGsi+1CUbt2rUT6en
pIiUlRbRu3Vq0b98+4GEHTIJOHnYb2dHUJMSuOS7xXQcNRxDFMxqn2WMEKpUlQevJ7LUPEl10RmWdnaKOgdcwYd8
VZh+TQmru35IQ6vqzXnnllYjPFxcXxxGOmUNjYyOysrLQ0NCAtm3bGr07pDO7VIKuqADeu70Cf/ymCIAIbN71/ak
eSxeB2+1tIVGaixHGFJThdUyRfc7lSsAf777jCNed50ugrakx9gNQURE5+Vvrbh4llZZ/OjfuL2qwYbPD8t8Vsi8
192/VAVAyYABEVlNRAUy6zo0a5KEbDsv3bcdzg1+2DJg0SX6bCn+FjEQl1mFkwLLcXG9XZEJ6LqqqvN1d0VRWGj/
URy4ocTr1S/52u4EnngAefjj0Ob0CLyIdqLl/qx4FBgAejwdffPEFjh07Bo/HE/DcpZdeGssmiShGvkFaw7EBTug
wgqiiArj3XvnncnOBZ57xPh+mhcgDCYeRiw0oQG4ucNttwDnnGNCCYKXaB4omNNOQwwE89BBw/vnJNeqOkprqAOi
jjz7C1KlTcfDgwZDRYJIkwW2W0RREScI3SGt4hJnLA6i5wfu6Y8K18jz7rPd5h8P7b1CLkJAkSAI4WLoAH4x3GNt
lYrXaB0YUnUl04EVkINUB0B133IHBgwfj3XffZSFEIhPwxTORZi4PoPQGH6n+D+ANdu69F5gwIWzNHumn1oMCM7Q
eRKsro6CInl3yxSJK4mp/SXF9yU91DlCrVq2wc+dOnH322Xrtk+GYA0RW4kttSYEbB5CHbqjzDz9vTkCC5FSRAxR
LzozZ7yBxJBjLpeUkNIeJdMXraw9q7t+q6wBdfPHF+OKLL2LeOSLSlq9hQ0gOlMBb3MiDwJZZDyRvSV8FBVt8JYQ
2u2LoUtO6Zo/WwtWViVJEzxc3BQ+U8s0EYoYSQhQ7Xt/kpLoFaMWKFXjwwQdx//33o3///jOH
YAkRW07xh41pRgYUoCUiI/iHbiZYvLYj6p2zzv4JHoApVsMioKbVUtFQZNXo+1sY0szfCmY1VqiOQMroOg09JCW0
0kiQJQgjbJEEzACIrah68pMCNAmzA+R3qMakkB5fOjn4XDM53jtal5oGEekcuuvxQA0e6sjuDFW/ORoyej7U7ht0
46lmpOgJFp+sw+Jqamph3jIj0EziAx4GcnJGqWg2C85098HapLUcRPJACgiBfF9uv3Qtw92aHohtDuJuzYcPiFUr
06PlwA+983THheupifZ2WrBjgWqk6AmlMp2rUlsapMEgvZq3oH2m2iwlwiUMInFrjIJxiAlwC8B5LNEpnXIhnxg6
9KJ0JpLIy/vfyzYQR7j3CzYQR6+u0JDetjBmvZ7BEXl/Sn65TYfh89tlnOHToEE6fPh2vOc//7kGYZmx2AVGejB
z90R5uXde0HB8XWo5qEc9crABBfDA+6d9tK6BaDkWzZmx6HC0mUC0zBGJtTvG6G6ccK1PZryewRJ5fUl/unaB7d+
/HxMmTMCuXbv8uT8A/PWA7JADRKQ1M3RPRBKtNJAHjpBpLBSUzQFwplCjEkJ4t1ta6u3OM+KGI9eNs3ChbJ1HTWd
Dd7uBDz5Qtm5wd4yR3TiRykWZ4XpG43Ak5vqS+ageBl9SUoKePXviyy+/RMuWLfHPf/4T69evx4evvvqKG1FVVaXDLhJ
ZW7QbBOC9QRj5t4NvKL3SuqZqbgxqb7rNZ+xItIoKb2vAqFHeFrFRo7w/AzGNnlf9vo8/rmz94IDVyCLX0QJcI6+
nUjFWRyCLU90CtGXLFnz44Yfo45sWPHDj32k8iy1NwgjBplEumvYDlqpoeK9aa
b6KRTJa10Bw5on+QbbbaR5sK1umlQ5DpmdkKi5iwgyUd1AOR2u9G6dWsAQMeOHXHkyBHk5+ejR48e+Pe//43169
1VrlBhJnNAk6nd77TTp1iuzFEuzmHk8gpudR042gZpEabbaS5SK1uzQNYB9wY3ixfa+NP+Vpyr9Ni1JbmrU8GDiV
L4llAkpPaDOvhw4eLFStWlihg7dqzYuHGjuPHGG0W/fv3UbS6jFTzjQJTMhIsESOWghl
1jZS+L+A9J9FGVG253yXqHIFDseocuWLL/aEv1GrUlm8EWrhrq/R6NjUJsWuOS3zXwYJDycg01Ny/VQdA77//vnD
99GHct2+fOO+884QkSaJjx47igw8+UL1Wh1Q3C6uRuuHLnQpISf78rK1MWhCgZ8q/H+z74oILPx09Rpif
oxR6ZkxquLEGs5z9cgKt0ey6XELdmu4QbknCb5UNBlpSQYfDN/fvf/0b79u1x//33o3///jOH
CL73k7RrzcTqV5xZpSath5Go/m5oNX1cxn4MbDl2mfpAr86DkelZUAJOuc6MGeeiGw/IjczgWnRRSdf/WPRyL4Mc
ffxSzZ88WeXl5IjMzU/Ts2VPMmTNNHuN3usK9xuVxizJgxomPHjqJNmzZiyJAh4v4v333w9YZ/HixQJAyOM///mPov1
iC5B5WbXYmhDy+66kW8MUdKrgaJbCkFq00sXy2dSsdVBFH56e3X1qr6fv+EdAx52ipKLm/q06Cfr777/HvHnz8ME
HH+DYsWPeDwBz+/fv1/XX3wkskn0a5dOyx
evBjXXHMNtm7dikGDBnDGVpoojpebM0Opi9lo60SRylImm10vHCo5mSTqNtxZMrJ9NzWrQqMi0V5p

rH0tSvtrr6RshOVzpXhk9UoBsRXUAdOutt2LdunWYNm0acnJy4ur22rJlC8aPH4+rr74aAJCXl4fy8nJs27Yt7Gs
WLFgQ8POTTz6JlStX4q9//WtAACRJErp06RLzvtEZZqhgbPViaz5yNwitg0tNr5fVo04Vwo2CizbkP97PZqzv29y
6z3MwIvpqcJ+VgxyFn61EjMLzxTP1MLCQESUvtc1LWVlZYuPGjTE1TQWbO3eu6NGjh9izZ48QQojq6mpx1llniTI
VmYZut1s4nU7x/PPP+5ctXrxYOBwO0b17d9GtWzdx9dVXi08++STsNk6ePCkaGhr8j9raWnaB/UTrZMlYWW0klVJ
ad+lper3MMMFUAgR325w6pa4bR6vPZqzdgU1NQrRr0yQOIVe4Id+X5oYkDsIpKtc2mSop33fuUhB5/z2wx2eN9Kf
rKLC8vDzx2WefxbRjwTwej5g5c6aQJEmkpqYKSZLEk08+qWobTz31lOoYP48ssv/cu2bNkiXn31VVFdXS3Wr18
vrrvuOtGiRQvx+eefy27j4Ycfls0ZSvYAyEz3P6NG6ehJ6+BS8+ul8M6+9sFKU03sqoYWAajRn805c7zbnwDfKKr
AD5Vv2QS4/PsQ76gtrTQPxiLtv4ejwEghXQOgV199VRQVFYnvv/8+pp1rrry8OTm5ory8aefir/85S+iQ4c
OYsmSJYpeX1ZWJlq2bCnCnWrFkTcT232y0uuOAC8etf/1r2ebYAyTNTq4uZ9kULegSXmp8jhXf261EmGzgoatEwMAt
aqwDUyM9mU5MQHTqceY8JcIlDCPxgHYRTTIArZB/MkpTfPBiT2//vs60yUoDMQPMAaODAgWLQoEH+R5s2bUTr1q3
F+eefH7B80KBqnY0NzdX/O53vwtY9thj8/PyrPFixYtxDvvvKPovW699VYxduxYRetyFJiX0X/ZNme
mZnst6HHT1Px6KdzJEagMCRwUtazoNKRPSUylZQBq5GdT7hKloEmMQKW4HmViBCpFCpoEIESnTqH7YJZReM0/Cr7
9v6tDmVg3x8CdIkvSfBTYtddeq0P2EfDDDz8gJSWw6oPD4QgZWWasvH77+/fvHtb/
JxsgJFoPZbcZmPabHUHu9oiVfu4cV4EtHLrq465ACEbIdDyQcRi42wDvBlBDea3H77cC//x14jYCgvGmoS65Wmii
uNAFcy/nZjPxsyn0+PHBgHUaGLH/hBfkpNMwwCi9whKQDOTkjLVHfiyxO/3gsvOLiYtGtWzfxzfxzjvviJqaGlFRUSE
6duwoZsyY4V9n5syZ9yZYtq0af6fy8rKRGpqnjhhhRdEf/X29/3HixxAn/Oo888oh4//33SFJTU8X
WrVsV7RdbgLzM2Opilmb7eOnRAqTmeilpfKmsVJZXouQ4m4mu9DDe3H77cC//x14jYCgvGmoS65Wmii
uNAFcy/nZjPxsyn0+PHBgHUaGLH/hBfkpNMwwCi9whKQDOTkjLVHfiyxO/3gsvOLiYtGtWzfxzfxzjvviJqaGlFRUSE
6duwoZsyY4V9n5syZYtq0af6fy8rKRGpqnjhhhRdEfX29/3HixAn/Oo888oh4//33SFJTU8X
WrVsV7RdbgLzM2Opilmb7eOnRAqTmeilpfKmsVJZXouQ4m4mu9DDe3H77cC//x14jYCgvGmoS65Wmii

```
i8NzUQ349u43wUqT59/ezz5S9xm61AOxKTd/a6dOnxfTp08W+ffti7Z6zBOYA2UvMXfbRkoN9iY+Rcjx0njjJ5RK
iUGZU1CHkikK41I8C0zO7NY7ciaYmISrXNonvOuQKD0w4DEnP4XBaipK17gHEMWRHzAFKKmqSz810nZOYrknQWVl
ZDIDIMuIaKaX0pjZnjvybJyJZ0uX6afROaDKrByrfQ8/sVq2GrJl1GFKsk7bqOBomLJcr7P55AOEGAkaCaXJqjTj
OeMUyCW3SRormoWsANH36dPHMM8/EtGNWwQDIPuL6w1zpTQ0IvTsoueF36iTE0qWx3xD0mF1ZrxuVli0kZhyGFMv
xGTWtelOTENnZYfcxeCRY3KfWitPHK2n91fMPG4qZmvu36hygs88+G4899hg2b96Mn/3sZ2jVqlXA83fffbcGHXN
E2ohrgJOahIfg/A4lyZJffQXccIP351iSqvVIyNQru1XLktFGzPgajYo8Krcb2P1EBfo9XARAIKAmrl4J081t2AB
8803Yp30jwVY/uAGO0SPjO7WJTgzXSiyT0ObmeisZm/F4SJbqAOhPf/oT2rVrh+3bt2P79u0Bz0mSxACITCWuAU6
+m5qSX4TBgYbaJMhYbghWmpxJ63kUzDYMKXjIWvMbfrMy/xUrHbjnbjc21pVAQIQO6RdC/2nVFX4eRvetB0bG8T5
Wnj5e6XfmwQeBvn3NEYSTajHNBRbusX//fj32kShmcc2/13zuGiWa/9JUO1zGd5NQU0PESpMzmXAiRM1FKYhUgUI
UFQE96wyuZ5Soz43eM+XqSemxjx7trcs0ciSDHwtSHQARWUnc8+8VFgJz5ih7s+a/NGOZxFLtDcFKQYUZJ0LUQ5g
qge7xhbjnbjcuFVW4Di5l29Kr5S5RnxsrtVAGs9J3i2KmqAvs3nvvxWOPPYZWrVrh3nvvjbjus88+q8mOEWmleS2
Z4PIzirrsZ88GXnrJ200lR65OTqQukWiU3hAUdruYJqiI+0IkgBY1m2S653Y/WoGNdSVwQkVeiV4td4n63MTb0qR
T/SxFrPbdotgoyaoeOXKkOH78uP//4R6jRo2KK3vbLDgKzJ7iGuAU6/DrWOqIqK0hYsZRUZFoONJM00FrcY5WCrs
vYUoVeIweSq335yaemXLNMnLMat8t0mcY/L59+4TH44lrx6yCARDJivWXoe/OuHSpEB07ajtsPfg9rFRnJU6a3iP
jrNkUdl/e9AYB4YKd4OUeJHgotd6fm1j+cDDbZKNJ+N2yMl0CoJSUFPHll1/6f540aZI4evRobHtocgyAKKx4fxm
atZBfnBJ9j9D0HhlnPaVI+zISlapa/zy5Fm5dCPchUPOHgx61rSip6BIASZIUEAC1bt3athWhGQCRrmzWrJ7o3gr
N75FxFGmMti9ToKyY5m/xK7FuTqV1b+zRPgRKI2SF12LHc5WWPVWkL10LIRJRnHyF/KqqvA/AmzRrpro2wcIkpBp
R507z+o9xjFaKti9HwkwqGuyCOdfh0odGKtsPs6moAK67LnR58Icg2sVwu4EPPlD0lvPvqcfGZ9TXDiVqTvEweEm
SIAUNCQz+mYgUWrkSmD4dePxx72PMGCAvz3szMZuKCu++jRoFTJ3q/TcvD+7lFRHr3AHqyhoppfno6jhGK0V7jw0
oQC1yg+s9+wlIELlOXDrbosOp3W7g9tvln1PzIfB9xh5/XNHb1iPHH1+Z8StD1qC4BUgIgenTpyMjIwMAcPLkSdx
xxx0hU2FU8NNIFJmVpgeIsK8pE4twIZbjMOT3VXVLjEKa1/FTMY2F2vfwwIESLIQL8sOpJQBYuMC6w6mfeCLitBq
KPgThPmMyPJBwGLnYgAIIoUExaSOH2pPhFLcAFRcX46yzzkJWVhaysrJwww03oGvXrv6ffQ8iiiDa9ACAPs0msVC
wrwtQihRE3let69wpqTHpcHinWlMkjiKNSurlbXMWwrMsfIVo0wS7arndyiulh/sQRPqMBfH81IpWigXwwHstmsd
XqoVp2WSTUhLRPyXJepgETbrRclZ0vSnc1xGo1PdQZBJow428Ck6GVpWIHWNyuuKBfXYbTq30sxzpQ6BiGwfhFBP
gkn26rEzlvpttqD1pRs39m1NhECWSSaYHcLu9+dfl5d5/ZRucFO5DV8ivp8lsAWH+Si9EBd58M3pvharGtDDTWER
roYkyBdiZl/sqRNtl7iiln9EOHcJ/CBRu41E8iJ6owYow3a2qimZbqRWWdGVoANTU1IQHH3wQPXv2RIsWLdCrVy8
8+uij8Hg8EV+3bt06/OxnP0NmZiZ69eqFF198MWQdl8uFvn37IiMjA3379sWKFSv0Ogwi5Uwwganiln+F+1CPnJi
m94oahPlyQ4KHWf2UK9Xns4qI96iYukdiDFJijJ2sTelntKQk/HlUuI1KjPZ3ezUXU5Bt5UlaSVsJaJEK6/HHHxf
Z2dninXfeETU1NWLZsmWidevWYsGCBWFfs3//ftGyZUtRUlIiPvvsM/Hyyy+LtLQ0sXz5cv86mzdvFg6HQzz55JN
i9+7d4sknnxSpqanio48+UrRf7AIj3cQzPYAGVLX8K9xX15tNqnuOotYOUlDs57tsp0hBU9TeE9XdI6RMtM8HIER
2duTPsoLP2PfZTuFAk3a1Q8uU1WbiB8eadCmEqIerr75a3HzzzQHLCgsLxQ033BD2NTNmzBB9+vQJWPbLX/5SDBk
yxP/zpEmTxNixYwPWueKKK8T1118vu82TJ0+KhoYG/6O2tpYBEOnHoGrQMRUQVLivatJbogZhbzYJ8dxzim5S0fK
PgtNP7JaGY7hwnw/fBVXyWVbwGdO0dqiV8vBINcsEQHPnzhU9evQQe/bsEUIUV1dLc466yxRFiHyLigoEHfffXf
AsoqKCpGamipOnz4thBDC6XSKZ599nNmCdZ599VnTv3l12mw8//LAAEPJgAES6MaAadMy/9zXc12hBWCFcos6hfPL
YuzqUhdw3U9AkRqBSTEGZmNipUjSdagp7GEbMr2k7Wnw+FGxDs+DV4FZY0pdlAiCPxyNmzpwpJEkSqampQpIk8eS
TT0Z8zTnnnCOeeOKJeWJgGWbNm0SAMSRI0eEEEKkpaWJ1157LWCd1157TaSnp8tuky1AZIgEN0fE1fKv0b5GCsImwCX
cMrOmR3qsm1MZ0HgwAS5xCKFRzpb7XRz0oyctPh+J/D7YdE4+stBUGG+88QaWLl2KsrIy9OvXD9fW4SiIrCGOgseaCReEFWADnIgwMqe
5oCFmhYXA+HFu/JhbAumr0EknJCEgIGEBSrES40NGEwkRuVhxRYX8fVLpPFSWCZ7iPVCr8c3JZ4mLQ3owdBj8Dz/
8gJSUwF1wOBwRh8EPHToUa9asCVi2evVqDB48GcVGlpaRHXGTZsmEZ7TmQ9cRQ81ky4ysk5YWoJyZKpoOzYvAGZXx0
OM+MWkAKB7qhFAcIPbZZrnQo3Ev/wYe/8n9GKBlui2LDbDTz6qPeAwpQcMNcOa8hutZlIHd075CIoLi4W3bp18w+
Dr6ioEB07dhQzZszwzwrzNz5kwxbdo0/8++YfD33HOP+Oyzz8SiYtChsFv2rRJOBwOMW/ePLF7924xb948DoMn+ok
B+dch7x+cfjEClcryfp57Tj43RGGC0/UoU5z8HS1hG4g8ytsSxYblPgxMCiYLs0wSdGNjoygpKRHdu3cXmZmZole
vXmL27Nni1KlT/nWKi4vFiBEjxMiRI8WYMWPEyZMnxdixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE
2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMn
xNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkT
J8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJ
EyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPE
iRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFj
xIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgx
Y8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SY
MWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPE
mDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGj
xJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahR
o8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8Wo
UaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPF
qFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFj
xahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNix
Y8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TY
sWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE
2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMn
xNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkT
J8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJ
EyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPE
iRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFj
xIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgx
Y8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SY
MWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPE
mDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGj
xJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahR
o8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8Wo
UaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPF
qFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFj
xahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNix
Y8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TY
sWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE
2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMn
xNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkT
J8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJ
EyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPE
iRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFj
xIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgx
Y8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SY
MWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPE
mDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGj
xJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahR
o8SYMWPEiRMnxNixY8WoUaPEmDFjxIkTJ8TYsWPFqFGjxJgxY8SJEyfE2LFjxahRo8SYMWPEiRMnxNix
```

VVSzstAHXvzgyYnygZ7HhuEflhZsOwunkdBCUFBgAkWX5/ogtKvIGO3Ijpw3tZrCyn2oLDDx8GOU/LZLLf7FCkKm
6pVBhxPT6c/VIURAj6DHFlWalHzgdBCUxdoGRpZm6m8Gqwgz7kst/AcwfZEZLdwnJ41EYMaV0M6bZS/NReZwOgpI
UR4HJ4Cgw67HMhJNmF2XYlwcSDiMXPVGDbk6HZXpKfEEDoKDGoomHcFlyVB5RAnEUGCUd/hGrEYX5L9uf26C4PI8
ZqGopNPEQLkuOyiMyKeYAEdEZCvNfBnauR5h8aHNyu1HYYQOunVePT7/Kwb86FaBLN0f4lsJwCcK5uYYmCNthVB6
RWTAAIqIzbFZbwO0Gdj9RgZ4LS9Dq34eRAmAggAtyc/HP2xbizfrC8F2mJkwQttnlITIUc4BkMAeIkpaJ81/Uqqg
A3ru9An/8pgiACOjv9/xU4acIy7EChZaZPNdGl4dIF8wBIqLYmDj/RY2KCmDSdW489E0JgoMfILCgYQrclpk81ya
Xh8gUGAARUSCL1xbwFXIejg1w4nDYX3K+hO4CbLDU5LkWvzxEpsEcICIKZcL8F6V8I6WGQ1kmcM5P6zUfQWX2yXM
TdnlYX4JsjAEQWZJpfi+bZkd0oEcJ4wTwjYCqh7JM4OD1rDKCSvfLo1m5aSJzYhcYWU5FhTcRdNQoYOpU7795eTH
kb7jdQFUVUF7u/Vdt34dmO0Ja8o2A2oAC1CLXn/AczAMJh+DEBgTO48ERVNCh3DSR+TAAIkvR7PdyvMELbxCm5Zv
6QkgOlMCbMRwcBPl+LsUC/wSvIVNiJCtfEpXcMDMrJUsRRcEAiCxDs9/L8QYvvEEEirclTWPNR0q9JRWiCMtRh8C
M4cPI9Q+BB2IfQWWyQ9cGy01TkmAARJahye9lLYIX3iDOMGk3YPORUitQiDwcwEhU4lcdyvDi5EoUdKsJmNU+lhF
USg7dkgESy01TkmASNFmGJr+X1QQv4TJMeYPw8rWkBQeTvpY0g8dkB46UciAnZ6Q/R/22OHPXlRw6YNEcYpabpiT
BAIgsQ5Pfy1oEL7xBRG9JkyRvS9r48YaOigs3UiqeEVRKDv3224Fvvgl93iSxYWS+JKpo5aaTPlmKrI5dYBZmyeb
1OPh+LwdXwPVRlMSqRfCiyY5YnIW7AeP93ig5dLngx/ccYPIUMZabpiTBAMiiTJp6oStNfi9rEbzoeIOwTFBr0W5
ALb438R6SiWPDM1humpIAAyALSuYR2HH/XtYqeNHhBmGpoNaC3YBafW+0OiTDYkOlUXZhIXDgAFBZCZSVef+tqWH
wQ7bB2eBlmHk2eN9s0OGa4JNlNui4CzDLVbl1Or3Bj5pf8BpVgg6XVOuLyUz3R7eaackBw6tla/m9iXboSlVWGlB
om9WdyebU3L8ZAMkwcwBUVeVtGYjGkF+uVmOSaSwsG9T6ojYgMBJoHrUBprjhav29iXboHToA//539NgwodfTJFG
2Sb52ZFNq7t/sArMYi6ZemJNvKNCUKd5/9fgtrKC7wbL5xNG6AQHT9NVq/b2JdugvveT92TQ5xCYp3mmpbl6yPQ6
DtxgLpl4kL4XdDZYOasNNSw5472wmGSavx/cm2ozsy5fLX361vaya0KL+VZxMXjaKkhADIIthiQ6LUPHb3tJBbbj
+jKoqw2+4zen1vYlUTyhagJRQBkfZFikbRUmGXWAWwxIdFqCyu8GyZYUi9WdocMPVsiSAUd+bRPSyKmJwlG3Zbl6
yNQZAFsQSHSan8re9JYPaaGPK9+5Vtp0wN1w9ckWS+ntjcJRt6W5esi0GQBbFEh0mFsNve0vdnJW0cL38csw3XD3
rXCXt98bgKNvS3bxkWxwGL8PMw+DJAuIYc22JIcJKj2/OHOCRR7z/DzdMPijysGxJAKvQqv6VSmrKRvG6Ujw4DJ7
ISHF0N5gmZyQSpS1c55yjulnLKrkilpmyJJhBTWCW7OYl2+MoMNKWJZowdOb7bV9U5P3tLtf6YeXf9mr6M0aOVDU
Uygq5IpYvphxp6JqOfN28pikNQEmPXWAy2AUWI8vfGTRmUHeD7nTszzB7pXOTFFO2NP6NRHriVBhxYgAUA94Z5Nn
1t72SaTBiuN5mzhVhfpI12fUrSPIskwOUl5cHSZJCHnfddZfs+tOnT5ddv1+/fv51lixZIrvOyZMnE3VYycckZfZ
NyQxJPUoTVtQktug0bM2IXBGlh22V/KR4WTa/SQan3qCIhIGOHTsm6uvr/Y81a9YIAKKyslJ2/RMnTgSsX1tbKzp
06CAefvhh/zqLFy8Wbdu2DVivvr5e1X41NDQIAKKhoSGOo0silZVCeH//R36Eua6kI5dLiNzcwOuQm+tdHst6wZq
avNe1rMz7b1OTbrvtdEbfHS3eJ9xhl5Up+5iXlWm7j4kU68fAjFwuISQp9PpIkvdhxWOi6NTcvw0NgIKVlJSI3r1
7C4/Ho2j9FStWCEmSxIEDB/zLFi9eLLKyslS978mTJ0VDQ4P/UVtbywBIjWS4M1iR0juASe8UOsVWfmoP2+5xvkk
/BjFpagoN5IKPyenU/jNFxlMTAJlmGPzp06exdOlS3HzzzZDCDR8OsmjRIowZMwY9evQIWP7dd9+hR48eyM3Nxbh
x47Bjx46I25k7dy6ysrL8D6fTGfNxJCVWOTMfpd2Sp0+btvtSz97DWHptLTtliQJ268VOlu5Kio9pAqC33noLJ06
cwPTp0xWtX19fj/feew+33nprwPI+ffpgyZIlePvtt1FeXo7MzExccckl2BuhNP+sWbPQ0NDgf9TW1sZzKMnHznc
Gq1J6B/j975PyThHLDdLOtWzsFjBYoZwCGc80AdCiRYtw5ZVXomvXrorWX7JkCdq1a4drr702YPmQIUNwww034II
LLkBBQQHefPNNnHvuuXj++efDbisjIwNt27YNeJAKdr4zRGPWjFGlv9n37dN2exYR6w3SUlOWqGC3gIGN0qSEKQK
ggwcPYu3atSGtOeEIIfDnP/8Z06ZNQ3p6esR1U1JScOGFF0ZsASIN2PXOEImZh5go/c3eu7e227OIeG6QdpxPzG4
BAxulSQlT1AF65JFH8Mc//hG1tbVITY1enLqqqqqjRo3Crl27cP7550dcVwiBiy66CP3798ef//3798ef//xnRfvDOkBxSJa
iG2ave6S0oM4XX3iDIDMW3tGRmesNGcGO50OnUlVkcpapAwQAHo8HixcvRnFxcUjwM2vWLNx4440hr1m0aBEuvvh
i2eBnzpw5WLVqFfbv34/q6mrccsstqK6uxh133KHHbMVAzZqh7ozcrZIwq7ZZMT0/K7stk7rWVY8fzkYyN0qSO4QH
Q2rVrcejQIdx8880hz9XX1+PQoUMBykoaGuByuXDLLbfIbu/EiRO4/fbbcd555+4/fbbcd555+Hyyy9HXV0d1q9fj4suukiX/ac
kZJWMUaV3gCS9UyTpYYdlx/Nhx+5K0o4pusDMhl1gFFF5uTfnJ5qyMm9LmNGUdksmS/dlENMdtsE7ZLrzQaSCmvs
3Z4MnUstqGaNKZ/+Oc5Zwq944DZocXZ4JJhQ21fkg0hFbgGSwBYgismPGaJxMcN+OjxmiN7Mn1sfADKeVkoulkqC
JLMeOGaNx8N23g9Oi6uq8y81QFSAiM5QzsEJivUpmOK1EkTAAIoqFHTNGY2D5+7ZZojerJNYrZJbTShQJAyCiWHG
IibXv22aK3mxUitlMp5UoEiZBE8UjyTNGLX3fVhO96X2NrZZYH4GZTitRJGwBIqKYWfq+babozUZzN5jptBJFwgC
IiGJm6fu2maI3GyXWm+m0EkXCAIiIYmbp+7bZojebJNab7bQShcMAiMik3G6gqspbeLqqyrxJo5a9b5sxerNBYr0
ZTyuRHBZClMFCiGQ0KxYWtGzRO7mT7XR679JmPdkWwNNKRlBz/2YAJIMBEBnJhgWBzc+y0Zu58bRSojEAihMDIDK
Kb5aNcMOIk3CWDSIixTgVBpFFWbqwIBGRhTAAIjIR1lAhIkoMVoImMhHWUDERuyWw2O14iOLEFiAiE2ENFZOw21T
mdjseIg0wACIyEdZQMQG7TWVut+Mh0ggDICKTsWxhQTuw21TmdjseIg0xACIyIRsUBLYmuw3Ds9vxEGmISdBEJuV
wACNHGr0XScZuw/DsdjxEGmIAREQhknbAkN2G4dnteIg0xC4wIgqQ1AOG7DYMz27HQ6QhBkBE5Jf0A4bsNgzPbsd
DpCEGQEQEgAOG/Ow2DM9ux0OkEU6GKoOToVIyqqrydndFU1mZJMnZdkuEstvxEMlQc/9mEjQRAZAfCJQCNwqwATm
oRz1ysAEFqK9Pkpum3Ybh2e14iOLEAIiIAIQOBJqACixECZw4kxBUi1w07F0IgN0mRGRtzAEiIgCBA4YmoALLUYR
uCMyG7oY69HskGbKhicjuGAAREYAzA4ZShBsLUQJAhPyCSIGABCRJNjQR2RkDILI1t9ub3Fte7v2X9+zICguBD+d
sgBOHw/9y4PQJRGQDzAEi26qo8A7rbl7TJjfX28rBkb/hXXoOp08gIvtjCxDZUtIX9IsHp08goiTAAIhshwX94sT
pE4goCTAAItvZsCG05ac5prBEwekTiCgJMAAi21GamsIUlgg4fQIR2RyToMl2mMKikcJCYPx4Tp9ARLbEAIhsx5f
CUlcnnwckSd7nmcKiAKdPICKbYhcY2Q5TWIiIKBpDA6C8vDxIkhTyuOuuu2TXr6qqkl3/X//6V8B6LpcLffv2RUZ

GBvr27YsVK1Yk4nDIRJjCQkREkRjaBfb3v/8d7mZjkf/xj3/gsssuw8SJEyO+bs+ePQHT3Hfq1Mn//y1btmDy5Ml
47LHHMGHCBKxYsQKTJk3Cxo0bcfHFF2t/EGRaTGEh23O7+QEnipEkhFyWhDFKS0vxzjvvYO/evZBkapBUVVVh1Kh
ROH78ONq1aye7jcmTJ6OxsRHvvfeef9nYsWPRvn17lJeXK9qPxZGZGVloaGhISDQIiIyDZY6Jwqh5v5tmhyg06d
PY+nSpbj55ptlg5/mBg0ahJycHIwePRqVlZUBz23ZsgWXX355WwLCmzdvDru9dS9ePX9U6dOobGxMeBBRGRaLHVOFDf
TBEBvvfUWTpw4genTp4ddJycnBy+99BJcLhcqKiqQn5+PP0aNaNHY/369f51jh49is6d0we8rnPnzjh69GjY7c6dOxd
ZWVn+h9PpjPt4iCh+nMxWBkudE2nCNMPgFy1ahCuvvvBJdu3YNu05+fj4mlmQtv+/qXB7c
gCSEitirNmjUL9957r//nxsZGBkFEBmMPTxhqSp2zhAFRWKZoATp48CDWrl2LW2+9VfrZZuPrqq1W/dseOHchpVtJ36NChWLTcA6q1e
vxrBhw+LeTyLSH3t4omCpcyJNGN4F5vF4sHjxxYhQXFyM1NXB3Zs2ahBq5eXxno16+fP2na5XL
B5XL5X1NSUoJL70U8+fPx/jx47tChQ7j55ptDnquvr8e
hQ4f8P58+fRq/+c1vMGDAAABQUFGDjxo149913UdgsIWDYsGF4/fXXsXjYxYMABLlizBG2+wRpARBbBHp4oWOq
cSBOmqgNkFqwDRGScqipg61Kjo61VWJmkLkI9clrjT6Q1+kjpLnJKZmvs3AyAZDICIjON2A3l50Xt4amrYyMFK0ES
B1Ny/Dc8BIiJqztfDU1TkDXaaB0Hs4QnicR5MxhR7AzPASIiCsbJbIlIb2wBIiJT4mS2RKbxeEepsSk6ScGTsSEivTA
ASiQmLBIREZkCA6BE4cRGREREpsEk6ETgxEZEREsmwgBIb5zYiIiIyHQYAOlNzcRGEREElBAMgPTGiY2SY2IiIhMhwG
Q3nJytF2PiIiI4sYASG8FBd7RXmFfEd3vtxEREYzYgpTTAcYAsmQsR5DItpPm9Fuu3R8bgFW7oO6dD1hxh2WTq0gaOEG4AU1HJFfGQmStF7fxQLO4yCgvxBwRvxp4nIsf4mHnCpoGtTjhZYt8VQhEp3YYPHkXLXtEyrmHknbmqK8DB6QXc6Ck/sYKV4TYrmSi3x3eyYgGmMz
wEGKicGIjIiIi02AAlEic2IiIiMgU2AVGRERESYcBcBEBERESUdBkBERESUdBgAERERUdJhAERERERJhwEQERERJR0
GQERERJR0GAARERFR0mEARERERemHlaBlCCEAAI2NjQbvCRERESnlu2/fnrd9eIw4sKOwQbR1WA7NtCSbbBHdu233wIAnE6nwXtCRERESnlu2/77uORMACS8e233wIAnE6nwXtCREREan3
77bfIysqKuI4klIRJScbj8eDIkSNo06YNLLtuYMlCnrmYOneEa7YP9DLn/MoDG7aqrXv9M2NrMjzd5DNkK/ux8jjsWUQGKjggeksiH
IgW+//RZdu3ZFSkrkLB+2AMlISlndbm27atT/Uzdn9GHl81mf3Y+TxWZ/djzFRxet5ceHSdBERESUdBg
AERERUdJhAKSjjwMPPPxOxOzeGfTOLAAnOAGmkEBAwxxDmrWkdKTKjsanNPqQ8ugjTGUxSxMQCIzVAEwb3KAf9wbvr5CF4n4mLMIQ8K1XzXJSiaT1hC9IzWnPCAk0p4I+LCYXyQM3p9CV47ZyFyRAY6BvQ1RSWQBGGz
AiIiKOkwACIiIqKkwwBIoUceeQQSiUzJeQSSJAU8unFvn96dOnh6wvSRL69evnX2fJkiwq4ON9xwA7
zs9GyZUsMHDgQ27dvV+pj/iadevj/+xkyMzPRq1cvvPPjpiiyHruFwu9O3bFvkZGejt9KGhjekjycmt9IN9xA7K
ll6FTp05o27Y5R/AAACOOnzdaw1dnWnWWFVVWJbv///rwvVWFWfZXMhcZLxhMXa6+o1tNr3MC8vT3f3Zf7qQqDnqRPb
9A9Qfo5m/gwyAVOjXrx/q6+doVdd+HChQHr1tbwOHDpg4cWLAem3btg1Yr7+q16ojHpmZOmXlrqS4Sppb4bsZm7fT59Q
uQVpaGt577z189tl3+f999j9s5+LnZJUH6JcGd6ryzJXD/YjbTdFgtE0NixHIPhLQHS+b+0vI/c6J4PiQS+LUPrq5pP1wO1Pwd7cJ+ZmnFP+NC/NYeOuJjebGnmoYGZNz7D2FHiK+p5IYDF/vYxKDbMBL+uFPhXiBrpGCmIqiw9K4tb/EI2cH83Ha49qKqhS3N+r61z2xhdZ1LyZ68yx4atKS+Jd
Eq30P//73vwfsw5ZHa99rPQBNQfn9W+g2qPz8cK3z8ftcdo6u+gIEUefvhhcccUJA8v8vvfQkG7DHKrBfAFJLYVQkgBAwf
8yxYvXiyysrLi3zkNPPDAA2L48OmQXgGjXXgzgzRp0+fgGW//OUvxGzAC/AdTHU8j7Y4ThfBTjZHrLxoIXgNYexoE2EPy
5Pjl9+/YVc+bMM8f9s9ttNgJSUlonfv3cx3BmRpvjJy17ANgdsE3sZgxNrHmnkP/BYNGQz0rfv3D
UXkMzfQfZAqTC3r170bvV/Ts2RPXX3ii/ixYtwpgp7R5+A5d999x169OiB3Nxcj6B3Nxcj
3MXjwYEycOBFnnXUWDg0ahJdjffjnia7Zs2YlLLL788yyVy1yBbdu24ccff4y4zuzB0H3377LTp
06BCw3MrX0GfQoEHIycyUYnB6NGjZ0+HZ0+fRpLly7F+Pfj2s8yB4h2b5gzv8ONqfm+Kzw/ZM
TyzU01Xcw4SGXRf3tb38Ty5F59++qlYs2+HhZnPfedNj+LbaZh+Sb4uuFl2aNNmFwFf4yWfeeeCNg+ZYtW8Srr74qqqu
rxfr168VVIIT7//HO9nRWrRIT4U6Lf3tf/eEtmJ5mDFc5MzKH6F+O78Mfeqpp/1bt7hRP7ZaqsRLWb+jg47cF9NRGWFIfOyM3bRDoG5GSCEZCqGg6
mBrnhMLlbmAbRBtexbwN/ll+9eU8XLoOXd7+Lz2/6+M2Izv8D/qDcQz/N9zRnp6V/s9dHTs/ezdfWqnBjxWB1+8ZQb68zBnFM0eZTbfM2zpu8lWc6aUWyT6F8JM
T7+OPPlxAG3mGDEBjX1jwY5jz5+S0y2Acn4UAh+SJkxd6Kl/6j7+x3wFb+T6pLsHTFyD/QkOd2tfqXl9HXcS6mZD/TqjUeUK1oKZmpPBBbjdc00+J6XUQiX4JJ3bRv7D91ns/AgE/T0NE5I3jy6dZWYzzu2enQHRs/
oZS9lQd4PgtZk9jdtHmqvH4mQPwKdrF57u7Xx0Hme6rTH+DL8lPUT9h1UEXNcXwrcx3hZO9qCWVKqCAOsTt5J1Frz+zt3v3fTL6MFwgLfVdKCG8F6m9EyBwKvj9HxdeUw1cWsaggyIyzHhHwLVp2Z0dXHdt0uxKJRZ07oltLyazLwXmVNb9omRJf++D3Rj0j5vvA9Wv6VVGDT/N5hGHTnbpFqO9VPOgOrjaclPiTuRbQcQpE3t1o4KmOiSfR0uY9e+g
h3RXbB1+mI4Kp4Ugx4jIfSgC05Zg/oR8TX5nSk6wJ6n6gVBdKk7FUvOQUuFyWDYsDUXsBeYFx+oTOvF5yMmHqX2mWR5WY2gy5qv2bS8SPe+fuYgxIXEKPGyHFWA6vDSWIj/1z1BPbbsujQD71Le0c8WKeCX6MuBWsVSUN8RtkyTWGVmIIsaUsXXeq9dIJsWmI0qYhNm8AgEOaQ6DLlPE6zO5d40
lYjk+nzfeeAO33HIL3nzzzTYwZMybiula7hnKGDBkSsPpl92uIZW+R76HDx4EGvXrsWtWT9acT0rfQebU3p8Plb5Dvq
oPb7mzPr9C6b2GM34HWQAFKNTp05h9+7dyMnJb9Fr+7EiLn0/LmsX1Juk0hBKqrq6Nu+XXHIJ9uzDs888b7a81lpu8hVwMNQU+CWW26Juk0hBKqrq
/D+mnbW7o0KH+EQA+q1evxmvxuDBg5GWxXLPWzRzp9eAcfPnQ2h/rqM6+OXrbsvAh0GtjB8QH73Dsk8jtW5hLWCL/DjXD9tsw26MxJgd3Qt/XVmxKp2gqOipvYHHzsHw5+KDXeW3ifLonKYOzMdk1/ip+qsaosTyRtCgvJUA+PHhOKMmVvbLyRXOY1XG5m4lGm58ejSKn
1BKzzPfRZvHgxzjrrrKifOST9B5tTenyAtb6DPmqOL9WtjrPSRdNVmXfhNSKLZ0SHeyH2/
bJ9q0aePPZJ85c6aYNm1ayiozOtuuOEGcfHFhHvv/+u85FHHhvv/2Ldvn8Lo8Cqux/aUrRbGf5qqh+60m4uaay2tyJ0Kqk/R4FLJ7RrV
pqeKJ54Qe/fuFa+99ppo2bIMu6Lp0qXX+vGfs3+aNmyxjpbTnnvEZ599Ji/H24GpM9tcvf/LVbMN4uXD/zMXwxvTYQdqAhIEpT0dvNyrBg5BtOoOiRMBq8qnx0XTvgVxVWXf13tr
ePLF7924xb948kZqFKj766Cb7/bRZmJMz5KSqVCmHoX3vtVmf2hZr2vqTgMWIMqzKQQc6xdHcBa3m6CF2uTIdtmbvdttm/z/8CvN3DwiVkV8pV0I34ZAz/9EZ/z0EPqL0N0FwQ05bx7VcwThqoT+uqbiKkYQBpVbg/nt0XtwYzxsaXtDDhKu0wMoikzmrb/+Tb9fUBoWdZcdVtf9JnXmpBDz1ZmfRMq2
pAAiXy+Vfx8rX0Mcq30aJMcj6t5zD6BPJNvw84ZYGxYrRmh1nz7r8GJK7L6WdThXwbGDQ/AZ+hQKy7Ptiiw4+yFZXBeeCUBAVvtwZRdMEKVHcoqTV+c/z5+h9H9ssrRFtOWfjJp13bh1WNFPXTAPGdmkmTYpxQtAH6PyqnGqRzsQZlfcd99G0wEKdnlo7s94D4GfaEBqbuYcYUsfCvb8tqdCG46rS9RgdPH53QVD/OTqaM/xBzkuJ9PdP8CVa0Cnh
LSxNdu3YVhYWF4p///Kf/+eLiYjFAzjlcXAyE2Xf3pUbtLuvgGdPvQN3QdSu0deewmEWXxB3WNfmAmcgg66oxs77bpbsLmEpX9e0grN12bw3/8HYfV39+Pp5xBTXjrJmbQtEtMgFE+Jv3Fo3+/hrgiB04mLZAr7mUVdPRXzb25/WSOU3RX7KoC7PDY7NPiN0oFuGbfsaEYYTBt7R3d5f5DeyXG
i+utf/yrOP/98kZGRIfr06RROy33LHWFVVJQYNGiTS09NFl6e+MMf/hCy3WllJ6SgIiQn8iZGvTpE/DlTiS1xzdyRgGSrRrXW9yGwJR7Ss4KVv1Y5nl0RK3J2oa4cczdsQlkOiYoOOgRl3hL1xTrqkj7Dw9/+xgxY1a
ixAgBIORRXFzsX8fq13D+/Pmid+/eIjMz4d6Pvx314HtS0VYhA1FaWcPVJcpSY93g3mbyzZBOxjhtfMvetlPznje1ZsnsLMzmaRiRHTWG47ZuupiDHDLl1oNNU1FPZLH5Wx0QHeubrTbMUCdmAIgYC8YC6VbDHbq5Tp2Wxay1EKpz8eiEzQ6QR7CwxQ
z4ndQzfFZ7fvno/YzatbvoCSEEPq2MRERERGZC3OAiIiKOkwACIiIqKkwACIiIqKkwACIiIqKkwACIiIqKkwACIiIqKkwACIiIqKko6TA
AIiIioqTDAIiIiIiSTACIiIiIi1YXcORKlpaVG74YiS5YYS5YsQbt27YzeDSLyBJqYPn06rh32Wl22XVV
VBUmScOLEOLV22T0TJhwEQERUX22TI0PJcWSEM8++++Pvq1aq1Qr0L59+4qqGrR2XwFwkiv3iveYZ0PeVJjWxdxjyOQB2gc9++Q7pZ+v4WWBdtXq+lmBUzxdOAywRQ0uSAxvEB7dUTKPtEuLGkmpxmNNE7f6uMuacPL9E9Fi/h2AkGMAd15R7fD6dkcjkRJgwJDK6n+3X25zeTZ

9+vXD3/72Nxw4cACjRo0CALRv3x6SJGH69Olh32fTpk0YMWIEWrZsifbt2+OKK67A8ePH/c97PB7MmDEDHTp0QJc
uXfDII4+o2k9f19SqVatw3nnnoXXr1hg7dizq6+v96/haw55++mnk5OQgOzsbd911F3788Uf/OqdPn8aMGTPQrVs
3tGrVChdffDGqqqpiPLtEpBYDICJKiJSUFPz2t7/FP/7xD7zyyiv48MMPMWPGDP/zd911F06dOoX169gj165dmD9
/Plq3bg2n0wmXywUA2LNnD+rr67Fw4ULZ96iursbs0bPRr18/bNmyBRs3bsQ111wDt9vtX+eVV15Bq1atsHXrVjz
11FN49NFHsWbNGzK6+8giVLlmDJkiX+52+
66SZs2rQJr7/+Oj799FNMnDgRY8eOxd69e2M+x0Skgu7zzzRNRUiguLhbjx49XvP6bb74psrOz/T/3799FPLII7L
rVlZWCgDi+PHjEbc5ZcoUcckll4R9fsSIEWL48OEByy688EEyy688ELxwAMPKN7PxYsXCwDiiy++8C974YUXROfOnf0/Fxc
Xix49eoimpib/sokTJ4rJkycLIYT44osvhCRJoq6uLC9Ro8eLWbNmuV/n6ysrAhHS0TxSDU6ACOi5FBZWYknn3w
Sn332GRobG9HU1ISTJ0/i+++/R6tWrXD33Xfjv/33HQbnnnJmF7RPXqkgfBjnNbZoxbfSSfm6naegLSMt3jsYX6ss
2y7/8Ht7nwlqD74DT6KP1mUA3z5w8f1h9R/B98w7y/pxWt1+sC1hIIBhOFIhRrLS5hDLzsM0SWXWxYWGs/PqmvegSN
OHVO8nwDQsmVL9O7dO+w2AKBfv35w/cVPSrZ8a7zj1yG3Ygb33+ObcWQUgyl9YgMCtV9oFqSnbaS6KgqdHOY+fC3qQz8pw
BVMyz3wwwqsMx4kPAXTmvyZhbHJ1+K2EG3dVAs8tK1N+ePz4+bsD3PzPPbdg4NrSXTUnGP3FXkPLf/mm+D6MkwV5
llVU4//zz4XK5sH37drzwwwqsA4M+LufXWW7F//35MmzYNu3bvtu7btuwDBg/H888+rep8WLVpXEXScTLS3gZ0mS4PF4FO9
nuG0IIRS/j8fjgcPhwPbt21FdXe1/7N6l7dGPqwjPPPPIMhQ4bg3HPPxZEjR0LWczqdcVIx48JvHmbkv7RZ+ZzNPqr
OO1BRUYH77rsPGAgPT0dAAJyeeQMGDAH77Ga9BgwbB7XbjLFjOPvsswMeUtsVtwV0UMXXbpoFBg/bB2yfzpOrkVS8vawi0eCGxJ+fT
NDQ2orq4OWNahQwf07t0bTU1NeP7553553HNNdHdq06ZE2TE4ZNo9vMcPTJp2RYYFqDTH1dyDG3cBu4d8Xh8kwpbsC3MNaWx1F5YotQfQ01uWm6p8O5q4ZlBO8oPRz/jKq+H6dUkYjCmSpMFaFtzl
IkoR33nkHV111FVq0aIHWrVuHvP+sWbPQv39/3HnnnbjWKaDp0zHaQIMUfyiEYc5ae6kwqj3PmvnubnbeepsnyunpKzxrvK5q0Ntbq0XbxHDt+Tdru/Q8dGlHkiaJpEDKbk1sw2p4qbdawXr/9wDgRcGQ4ADtbhT9oLfUKJ/9mjxvt83/+D
GG2/EM888g0GDBuHrr7/Ghx9+iP79++Oqq67S/S/D2JKBBbgIhIM1VVVRg0bFDA46GHHsLAgQPx7LPPPYv78+Tj//PP
x2muvYe7cuQGvdbvuQOuuu3Deeed7NixyM/Px+9//339XzAQLdu3TBnzhzMnDkTnTt3xq9+9SvZ9z/9/Ghxvxe+vVq7Ny
5ExddBGGDh2KlStXIjVV2d96SvZTK4sXL8L8aNN96I++67D+Pf5z+fj5z3+OrV+OrVu3wul06vJ+RBRIEsEd10REREREQ2xxY
gIiIiSjoMgIiIiCjpMAAiIiKipMMAiIiIiJIOAyAiIiJKOgyAiIiIKOgyAiIiIqkkwwCIiIiIkg4DICIiIko6DIC
IiIgo6TAAIiIioqTz/wFGC8aYgdiDQAAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plot_scatter(trials_logvar[1], trials_logvar[2])\n"
]
},
{
"cell_type": "code",
"execution_count": 29,
"metadata": {},
"outputs": [],
"source": [
"from numpy import linalg\n",
"\n",
"def cov(trials):\n",
"   ''' Calculate the covariance for each trial and return their average '''\n",
"   ntrials = trials.shape[2]\n",
"   covs = [trials[:,:,i].dot(trials[:,:,i].T) / 200 for i in range(ntrials)]\n",
"   return np.mean(covs, axis=0)\n",
"\n",
"def whitening(sigma):\n",
"   ''' Calculate a whitening matrix for covariance matrix sigma. '''\n",
"   U, l, _ = linalg.svd(sigma)\n",
"   return U.dot(np.diag(l ** -0.5))\n",
"\n",
"def csp(trials_r, trials_f):\n",
"   '''\n",
"   Calculate the CSP transformation matrix W.\n",

```
"    arguments:\n",
"         trials_r - Array (channels x samples x trials) containing right hand movement trials\n",
"            trials_f - Array (channels x samples x trials) containing foot movement trials\n",
"    returns:\n",
"        Mixing matrix W\n",
"    '''\n",
"    cov_r = cov(trials_r)\n",
"    cov_f = cov(trials_f)\n",
"    P = whitening(cov_r + cov_f)\n",
"    B, _, _ = linalg.svd( P.T.dot(cov_f).dot(P) )\n",
"    W = P.dot(B)\n",
"    return W\n",
"\n",
"def apply_mix(W, trials):\n",
"     ''' Apply a mixing matrix to each trial (basically multiply W with the EEG signal matrix)'''\n",
"    ntrials = trials.shape[2]\n",
"    trials_csp = np.zeros((nchannels, 200, ntrials))\n",
"    for i in range(ntrials):\n",
"        trials_csp[:,:,i] = W.T.dot(trials[:,:,i])\n",
"    return trials_csp"
]
},
{
"cell_type": "code",
"execution_count": 30,
"metadata": {},
"outputs": [],
"source": [
"# Apply the functions\n",
"W = csp(trials_filt[1], trials_filt[2])\n"
]
},
{
"cell_type": "code",
"execution_count": 31,
"metadata": {},
"outputs": [],
"source": [
"\n",
"trials_csp = {1: apply_mix(W, trials_filt[1]),\n",
"              2: apply_mix(W, trials_filt[2])}"
]
},
{
"cell_type": "code",
"execution_count": 32,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
```

"iVBORw0KGgoAAAANSUhEUgAAA/UAAAHUCAYAAAB71AhDAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABUjklEQVR
4nO3dd3RVZdr+8euQnkASkwAhtNCL0kYEQhGwhCrCWAbQUHx/SBEp0QQGiAwRHQHEsOEqxEVBA3hmKiBjBoYjSghA
bMYAEAgMh0hJqCMnz+0NzXo4ppB82fD9rsZZn7+fZ971PNpjr7HJsxhgjAAAAAABgORWc3QQQAAAAAACgeQj0AAAA
AABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAiyLUAwAAAABgUYR6AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0
AIJeYmBjZbDYdOnTI2a3c0A4dOqRevXopICBNptN48aNc3ZLDnJ+jrt27SqzGtHR0bLZbDp58mSZ1ShPBR3733/
/vWw2m/bs2VP+jd1i1q5dq+joaGe3AQCW4AsBgAAsKrx48drx4d44d3uCDDxQcHKxq1ao5kYUoeXLl6t0nTpq1aq
Vs1u56a1du1r/Zvv/02wR4ACoFQDwC4Vy6dEmenp6y2Wwl2s6PP/jvqXTGG5o//fvqXTGG5o//73v/XQQw85uuw0AABx
w+T0AoNA++OADtWjRQp6engoICFDvdWjRQp6engoICFDvdWjRQp6engoICFDvdWjRQp6engoICFDvdWjRQp6enEQVR

K9guS8zP75z//qXbt2uU5Jr8PXKTfvsqub9++DmeXK1eurK+//lrZ2dn5BvvAwEBdvXpVv/76q0OINsYoJSVFd91
1V3F2J19Xr17VqVOnHEJzzt+7nGVl0VNQUJCaN2+u6dOn57k+5wMxAEDZ4PJ7AMB1hYWFycvLSx999Hj2r
Dhg269957JUmNGjVScHCw/vd//9dhXHJyssMTwXPGtm7d2v2v7n2rPjAwYMkKenp2JiYhQTE6Pq1asrPDzcvt5ms8k
Ykysgv/fee8rKyirRvt57773au6ev3evdu/e7bB80aJFstls6tq1a5G32bt3bxlj9N///tdhn3P+NGvwZTNL/ffDwx/2
aP3++w2svLy917txZ//rXv/J8KnpaNiwoerVq6cPPvigwG8KKKy89s0s0Yo3fffbfE276eDh06yN/fX3v37s3z/W/
durX97PcfHTlyRHFxcblu3+jRo4cuX76smJiYf0vm/L3449+b5cuX68KFC/b1pWnx0Or5csWSLpt0v4y6qn3r1
768cff1S9evXyfG+LE+o5ew8AhceZegDAdfn7+2vy5MeqTp3StGnT5OonpqalTp0qSKlSooGn
Tpmn48OF6+OGH9cQTT+js2bOaNm2aqlWrVuCn+s169fP8XExOjs2bN69tlnHeb6+vrq7rvv1iuvvKKgoCCCFhoZ
q8+bNev/99+Xv71+ifR0/frwWLVqkXr166ZuvlVvDt2rX12Wefac6cOOo5cqQaNmxYQ5zOjeX6cfopDa/3KjzvEJmf4PQgiE9fjjQ3a9MdqfXRiKKR6Xe9bFLTuKMmUqWelx6IM4ykmrt0QHzwdKmzOLaa8Jht9WjI6JsY4GSEPOgTlhDI7dW1EJAAHQZRT+XO5bnutGkLy4Fw
apb///e965ZVXdPToUVWqVEl16tRRRS9+7ddtttxV5mwMHDtQ333jOXPm6IUXpAxRnkkJJSXk+7wIABnU2Y4xxhM
AgJvb2bNn1bBhQ/Xt21fvvvPOOs9sBCiUlUJUXVq1fXq3Wr9MAyzi7nQINGT8//73X+/GecMzcdzwIAKGecqQcAlKq
UlBRNnz5dXbt2VVBgoA4fPqzXXX39d586d09ixY53dHl6BgjcwAoVR4eHjp06JBGjRql06dPrPmqMjT8anbaWiuVcnLg6dOAeX6ESMl
bW+3atd8efN0++23h+AF2y9AAcAD88mXOqA9nnmd5Nnxt7WbqMk7wAdkkQnYzxNqE6dOvL09NSD2FHk7akAgVNQmDv+8+bNnuBQs8+Xp6G

Jr7STpICAgAKfWi/99uC8HKGhobLAg/0BAAAAACg2y5ypBwAAAAAAjgj1AAAAAABYFKEeAAAAAACLItQDAAAAAGB
RhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwAAAADAogj1AAAAAABYFKEeAAAAAACLItQDAAA
AAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwAAAADAogj1AAAAAABYFKEeAAAAAACLItQ
DAAAAAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwAAAADAogj1AAAAAABYFKEeAAAAAAC
LItQDAAAAAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwAAAADAogj1AAAAAABYFKEeAAA
AAACLItQDAAAAAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwAAAADAoiwT6s+cOaOIiAj
5+fnJz89PEREROnv2bKHnDx8+XDabTW+88UaZ9QgAAAAAQHmyTKgfOHCg4uPjFRsbq9jYWMXHxysiIqJQc1etWqU
dO3YoJCSkjLsEAAAAAKD8uDq7gcJIStEhystrqt20qS3nzqXFqtNN3hl5pPXBUqXfHqXhKvh/frrrZt20cCBkhZ1Jy7UVRIw+UgOHoyFyJ6ihgmLd4+vtRgc+i+yFKLgGc4yPbJ4DfI37hjzbVR
N9eGHHzqMSUxMVFoxHkIAAAAAAIAVWeJMvSQFBAToo48+KnDM9b6dj/voAQAAAAA3E8ucqQcAAAAAAAI4I9QAAAAA
AWBShHgAAAAAAiyLUAwAAAABgUYR6AAAAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAAABZFqAcAAAAAwKII9QA
AAAAAWBShHgAAAAAAiyLUAwAAAABgUa7ObgAAAAAAACswpoCVtnJrwwFn6gEAAAAAsChCPQAAAAAAFkWoBwAAAAD
Aogj1AAAAAABYFKEeAAAAAACLItQDAAAAAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkWoBwA
AAADAogj1AAAAAABYFKEeAAAAAACLItQDAAAAAGBRhHoAAAAAACyKUA8AAAAAgEUR6gEAAAAAsChCPQAAAAAAFkW
oBwAAAADAogj1AAAAAABYFKEeAAAAAACLItQDAAAAAGBRhHoAAAAAACyKUA8AAAAAgEW5OrsBAAAAAADKgzEFrLS
VWxulijP1AAAAAABYFGfqbyA2mYI/OQIAAAAA4BqOVXbGysG9XbGysYmNjFR8fr4iARAln/PLL+rYsaM
aN26sbvTZs26bvvvtPkyDZzQkgZPl6elZTl0DAAAAAK6V37lEzEz/fkpLS1N+xVT/kePhzBDq0Wf4Rt6ubWkB+3aFhYXp559YkH0vxrFmf4d93ZZ
/VqNGjfKc179/f/79fPfPbLS1Nvr6+kiTbdR6gUNA7WpK5153vvvvrUyMjKUkZFhes/fr5eni5JysZMVGZmpiTJy6vgfn8flqeSzL3e/Ez
ls7IU+i7p/GL1fc1Gy2q/qU1talOb2tSmNrWpTW1ql2/tAhW4YefYfb+35GA
3bNhQQ4cOVVRVRUVK45KSkpqlatmry9f9vXiii2N1XPPPaeNGzcCiRoSVtGnTci1fsmSJhjvDbzXbU
HAAAAAADIx8WLFzVw4MAb/0xWhXMj/0XFnKxFbHtv4G7HqRLd26t8LnxHbDP9b/NdI3YRaQWm9XVVfYYsJoxVxgwhNsbG+xwmX1F3X0p1+RJRIOXI
FRSkyMtL+Oj09XTr3/GTp6l9SRvT+4bs6HmYfJaQGIdRjW5dpgTsGEEVpP3GmeHjKd7zVHX0DFzTKYjcdeS8ZTzCZo7WJeQVZgTDf5w/JOVRjJ6qyCHrm7u+Tqj3ADMPf9tkbR8FS
/jSh/2mIIQ8X7Xeg80MFxGpF58kS+3NeqCKdWxYxI+VyjgGx3zyVq4ZMzb6gK4Urfw7CNUJLHY6uGPGKMa6jXJ3ZK66TjYz4WMBCtlOCZHVB2PEaiOS2L+KqdtxI9i5qhHv6sztUK5v6O9V
Z7Te1qU1talOb2tSmNrWpTe3yrV2gAjfs/NrFml+MyJ1XDs2PJe6p37Ztm/z8/OzuKLjlqeFmLuU2HV3E0fqZ/S4+v5mJszTVjJ+ANHVXHjhlYw4ZXyE55HiyaM1TqQgfqUUwMcLLd69vVq64YHakfhTlIAsQbzhd1v59
drc8++0wTJkxQt27dtGfPHtWpU0dRUVHq27dvvrUyMjKUkZFhf52pMO8NeKp8SpcrTU/Cvd3CvkR/Ez
ls7IU+i7p/GL1fc1Gy2q/qU1talOb2tSmNrWpTW1ql2/tAhW4YefXLtb8wszNNaXwcyxxpn7GjBiYnRvn37HJY
3bNhQQ4cOVVRVRUVK45KSkpqlatmry9f9vXiii2N1XPPPaeNGzeqc+fOedaKjo7o7WtGnTci1fsmSJVvL29S2e
HAAAAADIx8WLFzVw4MAb/0xWhXMj/0XFnKxFbHtv4G7HqRLd26t8LnxHbDP9b/NdI3YRaQWm9XVVfYYsJoxVxgwhNsbG+xwmX1F3X0p1+RJRIOXIFRSkyMtL+Oj09XTr3/GTp6l9SRvT+4bs6HmYfJaQGIdRjW5dpgTsGEEVpP3GmeHjKd7zVHX0DFzTKYjcdeS8ZTzCZo7WJeQVZgTDf5w/JOVRjJ6qyCHrm7u+Tqj3ADMPf9tkbR8FS
/jSh/2mIIQ8X7Xeg80MFxGpF58kS+3NeqCKdWxYxI+VyjgGx3zyVq4ZMzb6gK4Urfw7CNUJLHY6uGPGKMa6jXJ3ZK66TjYz4WMBCtlOCZHVB2PEaiOS2L+KqdtxI9i5qhHv6sztUK5v6O9V

16yZX1xJfwQ8AAHBdfG85AAC5Fevy+yVLlqhXr15ydXXVSy+9pLNnz5ZyWwAAwGoK83V6Vn3om1X7BgDc/Ip9T32
OGTNm6PTp06XRCwAAuMEVFG4JuAAAlL8SXztv+D84AAAoDSX5nYLfRwAAtyhuiAcAwGLIrzcYfiAAACcqcajfu3e
vQkJCSqMXAABQDm7VDFrwft+ibwoAwPJKHOpr1qxZGn0AAHBLuVWDNfLAwQAAKIFihfrbbrtNNlvu746x2Wzy9PR
U/fr1NWTIEA0dOrTEDQIAAKCM8IECAFhesUL9lClTNH36dPXo0UNt2rSRMUZxcXGKjY3VU089paSkJI0cOVJXr17
VsGHDSrtnAAAAAACgYob6r7/+Wi+++KJGjBjhsHz+/Plat26dli9frubNm+vNN98k1AMAAJQVzrQDwC2vWN9T/8U
XX+i+++7Ltfzee+/VF198IUnq2bOnDh48WLLurnHmzBlFRETIz89Pfn5+ioiI0NmzZwucc/78eY0ePVo1atSQl5e
XmjRporlz55ZaTwCAkn9vuTO/95zvXAcAAFZXrFAfEBCgTz/9NNfyTz/9VAEBAZKkCxcuqFKlSiXr7hoDBw5UfHy
8YmNjFRsbq/j4eEVERBQ4Z/z48YqNjdVHH32khIQEjR8/Xk8//bQ++eSTUusLAAAAxcCnaQBQKop1+f3kyZM1cuR
Ibdy4UW3atJHNZtPOnTu1du1azZs3T5K0fv16de7cuVSaTEhIUGxsrLZv3662bdtKkt59912FhYUpMTFRjRo1ynP
etm3bNHjwYHXp0kWS9OSTT2r+/PnatWuXHnzwwVLpDQBwayJ7AACAG0GxQv2wYcPUtGlTvfXWW1qxYoWMMWrcuLE
2b96s9u3bS5KeeeaZUmty27Zt8vPzswd6SWrXrp38/Py0devWfEN9x44dtXr1aj3xxBMKCQnRpk2btG/fPs2ePTv
fWhkZGcrIyLC/Tk9PlyRlZmYqMzNTkuTlVXC/vw/LU0nmUpva1Kb2jVjbmazaN3BTKOgvYFn/5Stp7StXij+3pLX
zm09talP71q6da0rh59iMufHPNcyYMUMxMTHat2+fw/KGDRtq6NChioqKynPelStXNGzYMC1atEiurq6qUKGC3nv
vvQIv24+Ojta0adNyLV+yZIm8vb1LtiMAAAAAAFzHxYsXNXDgQKWlpcnX17fAscU6Uy9JWVlZWrVqlRISEmSz2dS
0aVP16dNHLi4uhd5GfgH6WnFxcZIkm82Wa50xJs/lOd58801t375dq1evv3atfXVV19p1KhRqlatWp4P+pOkqKg
oRUZG2l+np6erZs2aCg8Pt7+Zfn4F71daWv7rSjKX2tSmNrVvxNoAUO4K+kerrP/BKmnt/OaXZC61qU1t69f+g5w
rxgujWGfqDxw4oJ49e+q///vgGjVqJGOM9u3bp5o1a+qzzz5TvXr1CrWdkydP6uTJkwWlSxQZGZnraff
+/v56/fXXNXTo0FzzLl26JD8/P61cuVK9evWyL/9//+/6ejRo4qNjS1Uj+np6fLz83P4hKSAzxEkFXyfZUnmUpv
a1KY2AEAF/4N5o/9jmV/vhem7pPtNbWpT+8at/Qd55dD8FOtM/ZgxY1SvXj1t377d/rT7U6dO6fHHH9eMWP02We
fFWo7QUFBCgoKuu64sLAwpaWlaefOnWrTpo0kaceOHOHUpLS7Pfw/9HOffAV6jg+IB/FxcXZWdnF6o/AAAA3IBu9OA
OAOWoWKF+8+bNDoFekgIDA/XSSy+pQ4cOpdZcjiZNmqh79+4aNmyY4aNmyY5s+fL+m3J9n37t3b4SF5jRs31syZM9WvXz/
5+vqqc+fO+utf/yovLy/Vrl1bmzdv1Dw8PnTt3Ltfy8+fPy93dvcRN5WXx4sU
aM2aMwsPDJUl9+vTRW2+95TAmMTFRadfcr/Dxxx8rKipKjz32mE6fPq3atWtXaDQ8Dw8PnPBQAAAAAUhFAPAAAA
AADcbzg7dMrinHgAAAAAAiyLUAwAAAABgUYR6AAAAAAAsilAPAAAAAIBF8aA8AAAAAMD/4SF7lkKox02vpP8m8W8
aAAAAgBsVoR7lgmAMAAAAAKWPUI9CI5gDAAAAuC6CQ7ki1AMAAAAAbgx8IFBkPP0eAAAAAACLItQDAAAAAGBRhHo
AAAAAACyKUA8AAAAAgEXxoDxYAs/LAAAAAIDcCPUAAAAAgJvDLXg2kMvvAQAAAACwKEI9AAAAAAAWRagHAAAAAMC
iuKfeYm7BW0QAAAAAAPkg1N9C+EAAAAAAAG4uhPpyRrAGAAAAgBuQRcMa99QDAAAAAGBRlgn106dPV/v27eXt7S1
/f/9CzTHGKDo6WiEhIfLy8lKXLl30008/lWXll30008/lW2jAAAAAACUE8uE+itXruiRRx7RyJEjCz1n1qxZeu211
OwcHBuv/++3Xu3Lky7BQAAAAAgPJhmVA/bdo0AgPjZ5+n5ycrPj4eCUnJysrK0vx8fGKj4/X/tZZ5+n5ycrPj4eCUnJysrK0vx8fGKj4/X+fPn7WMaN26
slStXSvrtsvtx48ZpxowZWrlypX788UcMqTJE3t7eGjhwoLN2AwAAAACAUmOZB+VNmTJFCxcuVFBQkL744gtNnz5dixcv
SRZKUmJiotG0vuV5gwYYIuXbqkUaNG6cyZM2M2rbtq3WrVunSpUlWvvAAAAAACUBZsxxji7iRtZenq6/Pz8lJaWV
QXgHP2ZMk8Y4CAAAAwC0mv6BYjICYVw7Nj2UuvwcAAAAAAI4I9QAAAAAAWBShHgAAAAAAAiyLUAwAAAABgUYR6AAA
AAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAAAiyLUAwAAAABgUYR6
AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAAAiyLUAwAAAABgUY
R6AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAAAiyLUAwAAAAB
gUYR6AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAAAiyLUAwA
AAABgUYR6AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAAAiyL
UAwAAAABgUYR6AAAAAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKIsE+4QXFYgTEvHJoflyLvHULS0tLk81mK/Dy/YyMDGVkZNhfp6enS/rtcv7
MzExJkpdXwXV+HwYAAAAuFXkFxSLERAzizDnlgn1ly9f1qRJkzRw4MACP+mYOXOmpk2blmv5unXr5O3t3eL5O3tLUlaurT
gWmvXlqhVAAAAAIDV5BcUixEQL168WOixTg310dHReQboa8XFxal169Zal169YlqpOZman+/fsrOztbc+bMKXBsVFSUiM

j7a/T09NVs2ZNhYeH2z8M8PMruF5aWonaBQAAAABYTX5BsRgBMeeK8cJwaqgfPXq0+vfvX+CY0NDQEtXIzMzUo48
+qqSkJG3YsOG69yN4eHjIw8Mj13I3Nze5ublJki5dKrjm78MAAAAAALeK/IJiMQKiWxHmODXUBwUFKSgoqMy2nxP
o9+/fr40bNyowMLDMagEAAAAAUN4s85V2ycnJio+PV3JysrKyshQfH6/4+HidP3/ePqZx48ZauXKlJOnq1at6+OG
HtWvXLi1evFhZWVlKSUlRSkqKrly54qzdAAAAAACg1FjmQXlTpkzRwoUL7a9btWolSdq4caO6du3rXp6e8YaP7/7/c
rHD16VKtXr5YktWzZ0mFb184BAAAAAMCqLPc99eWN76kHAAAAAFyXk76n3jKX3wMAAAAAAAAAEeEegAAAAAAALIpQDwDwA
AAACARRHqAQAAAACwKEI9AAAAAAAWRagHAAAAAMCiCPUAAAAAAAFgUoR4AAAAAAIsi1AMAAAAAAYFGuzm4AAAAAAAAD
LM8YpZTlTDwAAAACARRHqAQAAAACwKEI9AAAAAAAWxT31AAAAAIAiy8rKUmZmprPbsCQ3Nze5uLiUyrYI9QAAAAC
AQjPGKCUlRWfPnnV2K5m7++4OBg2Wy2Em2HUA8AAAAAKLScQF+lShV5e3uXOJTeaowxunjxolJVyVJ1apVK9H
2CPUAAAAAgELJysqyB/rAwEBnt2NZXl5ekqTU1FQVqVKlRJfic6NBmk1rn7eUFRVqVKlRJfi86A8AAAAAECh5NxD7+3t7eROrC/nPSzpcwkI9QA
AAACAIuGS+5IrrfeQUA8AAAAAgEUR6gEAAAAAN70uxbpo3LhxhR6/iat/a9fXy4uLkWaV954UB4AAAAAoMTK+4p
8Y8p2+8OHD9fQoUM1ZswYVapUSUOGDNHZs2e1atWqsi1cRIR6AAAAAAAAAucf78eaWmpqpbt24KQlxdjsF4vJ7AAA
AAMAt5cqVK5owaXVYVqbSWbWbZ2bZIkbdq0SZUqVZIk3XPPbLbzOrSpTz75RDabTzeYTabzT7e2Th
TDwAAAAC4pQwdOlSHDh3Sxx9/rJCQEK1cuVLdu3fXDz/8oPbt2ysMVGNGjXYsb3l7e2vYsGFKT0/XggU
LJEkBAQFO3ovfEOoBAAAAAC7o4MGDWrp0qY4ePWqfWn/ixIlKSkpSQkKCRo0aJX9/f9lsNvul9tepUrKUXF/frd2v8
M++sbBaEeAAAAAHDL2L17t4wxuqbDwxatisbvTwtPUAwAAAAABuGdnZ2ap2Z06q1atUtTX//UAWc8uvbtcn1q6aCw4KCzHfbu/Kpx5Ulf
FR6gHAAAAANwyWrVqpaaysLKWmpqpTp06Fnufu7q6srKwy7u5/DA0
Fxenll1/W2rVr850XGhqqQ7//XEgiOBaoY3TZPLsev8EeoBAAAAACVmTPn+KYkFCxZo0KBBeuaZZ9SUP16dN
HO3bsUM2aNfdqzDmpVSM9q1EitW7dW5cqwsqsMUUTvh1OCFmqDTu7cKhhNPGk7owCRAdbrmLPyebwlOcXr8aywm
b0eXLl5WUlKQ6derI09ud9MqsAAAAAAAMfl6kl2k0hQ9PSFyYVaMlwGIU
AAAAsilAPAAAAAIBFEeoBAAAAALAoQj0AAAAAQuMqAAAAAAABZFqAAAAAFGAAAAAAABgUYR6AAAA
xxhul3hsAAAAAwHpsNptWrVpvv6PGbnM6ynnorJMqL9y5YoeeeeQRj5sshzV61apR07digkJKQMOgM
AAAAAyGYr3z+l4Pjx4+rRbU++OED7dy5U7feeqt69Omjb775RpL5/nz59NnOzs76+PBj2Vxc3P1tyR6qmr
OAAAAAABWc+XKFQUQHzu7jRuu7Nzd1nq6JJDbtQLT+bhJpRhHiEsLtPLl+9n91euAxQ0ZSfd2Jrt9Hd4vc
x+zAAAAAAuKlkZmbKGPs7rCvvy8/WP967nnnH1kvf9+9aaXLsA+0qM8/Ls4t
dqtGjR+vnn3/WHXfXXfNHVJCoeeeee04PPPfSQvv32M4uTlFRUdq7TdR9Ljz+n59lMWq/X8/9+X40aNV
/fQ0ZMiTP/o0xyMPMlMIuLi8O6zCKEyps61L/88tyd/XnFXJ1VmDHLNhCj1n5syZ9H0v91X59aSt9bCdW9j7doitQk
AAAAAluDq6qrGs4GCdP39eV9X5jyvcVnT+58JyALayrV69q0aJDZ9vWDfu6K4gNRGJ6enpOnfPunP06P
7779f8+bN05Ej4kRzxzRPFu0vWL5Senqy6p4nlNmzJd8b64+Ek7fezD9vAPCBqSwXNf77vvq+tCcCjeDS+kEDPq
+vr559nrlyhVdunR2+lQufvRJX331lf0aEhJmq+0Pf2n93LcJmuuyatpnuRZL9Z+d+u8r1tumBjYI4BbnqZd/oKAa+EntbwrFcatlmKKTs11tYYzC
GRtpfp6enq2bNmgoPD5evr68keZv8kyTpn4m0quZ6dBBZS7vopxxdqK8SaSfYzBgD4Qbrd9hCtwyvEe+FvKJy5a1X0lRBdEPPC
u5eUlX19fLVmyRBUQVqNCCQvEnFUm0z4plK3Qw6hBD/+6k4dMVJRMSvwNHqHxbBDxmxNV3ef/cb0Z6rKSnoOMTV9OtcMvh4SFKIGEbEcUW2XOFV4Y8qpBHPTqyDs8CEmK/L6AbEEqUdZxfbJfQ24dgLbMkZC2wKsZ4N+
AAHl4eKhBgwYF9nP58mV5vvVezVeFuUDCeG+tGjR6enHrZJTVNP4v4Wb9vmNPwzcjfa7GT7d29W3HAqwJX/O/ASxrpFZq2Te7yGkqfXs7bw
M3njjDR06dCjPeR4HvLw8eiOi/XlKL1cGD1ZXRlJTvn96KbJ/ikD44vZpbUmbD3FjNa+MQjHD3LiQdU8K
zL/v371fz5s3twV2SSo6vVrZVPKZZ7jZWNZyZTAJ7H1DMW5ocCGaiuG13K89L+2tnrh7dLDaU6g8H3CwcOyUR8kiSrQZ9v3GHOb
goDLZdkREhO677z6HPpdd26jD6eFNVd66NY+J9tamV0b7QHKaOL6TXOhHFHUpt2H5XzlbQQQcB0RrMqfcRfAlKhbWjcGHdHY9ZT3Cqu
OTtbp06eVnJysrKwsbZSI6lh4+tFYsgTRiNQpe1MJ3yH5F0p7n5q7ZmXqR9rUoinr7hOSUq2ySr
Y48aNtXjxYmVkZNiv3t61eRt+Pu7q6ssrKzSbi9f1m+ilTqhVqaOnWqzp8/r1atWqlYb3JiYqLSuKE
dAAAAFBEAwcOVHZ2tp588kkDlJCjJnD7Jz+QOiHdHPn6vdyDlcTHxMRc9zvq87S4dVYoJlt0THXDDhx56DECtWrFDXrl119uxZLvV
0Ikd+99EDAAAAAErI4vcp+/r66Nnp+0NfVqyDn9w0RubIf9tUYQhh/fw9zwpWXGvwbFXdsknV8NAp5bYBVzjsAUSq0kGvXQSXrQkE
vtCstNnO9JHyLS09Pl5+fn9fl0+9WpB8uuxdz7ofD9oLg3HqJZqHfhL0HzmkvxCsI5zSoGvZtsnNHtXuQU6rSLk7a9KdOEu4PA0xqu2C3su
i5FDLnKkHAAAAAKAsLVq0SHXr11X8apXriY1HVnEyBtOnAx1sWzpbcmU0Jtbxjv9hI26nt3SwcKBmpRW8nuuSOkZHk8TvFjJjc+KKx15Ls6
HNH36dGe3VSBCPQAAAAAAkiZMFQAAAAAgCNCPQAAAACgSHJqIEqiLbgJRUM+BGrIqVnBw6yHPhkIrtDWzaKcyAEQJGHAGghpBHEAAAAAMhJFw
k6eLFiO7uxPpy3sOc97S4uKceAAAAAFAoLi4u8vf3V2pqqiTJ29vbkcvbmFs9tWkU1V6hb8eXubVbdbfJeVq3N4z/C9uC6hTdm4NllWBldqHd8eTcDaXd0rt3Ggq0uDDGcw4H4YN0HYE3o7LJv0d
qAQAAAACFFhwcLEn2YI/i8ff3t7+XJWAyhUBDaklOFXmNPD2+5NwUwTuLTp2bPunIgiYBn0pVU2M2zMHmP8QGdb6bBKzoOaDYrtP1PIaL9u+ySHYgivYLxNBuN6RFHqH/GWv5iUPHVGEuq9DFSxwGahz
4uJRaMEXx8aA8AAAAAASLiAPAAAAAAABFEeoBAAAAAAGAK2MAAAAAbdd5+fX4dFdbjKWwb0fzWMppp2zrWPHjl5SpUqy2wy6qdj5
dkyTVrFnTyZ0AAAAAG4lS586dk5+As8ay2goNf7PFw0EA8AAAAsvvgh1lcB6Qh3ZSTgL18tfr3x3B8ySzrc
caygvHGsoLxxrKKC8caygvHGsoLcYYnTt3TiEhpaoeC75jlTfx0VKlRQjRo1vjXv7goFxxrKKC8caygvHGs
oLxxrKC8caygN1ztDn4MH5QEAAAAAYFGEegAAAAAAAAALIpQA0x0IeHh6aOnWDW8nN0KbncIcayg
vHGtwBh6UBwAAAAARXGmHgAAAAAAiyLUAwAAAAAsilAPAAAAAAAAIBFEepLYM6Tp448UUBHAFepLYM6
3asuWLc5uCRB31Vdf6YEHHrAp8vuLZd31ybb9xdw7Vb9t2T5s2LfIsB2XrlRrLsN4Yo+joaIWEhtz/95JxmYwkzZ87UqqXaVa7/ZaZlYqVkzpFrVVVvqfjHmt1N9y1SdbxmbL
KlSrq27evEhMTE1Cf+/F4ffrsrf69fX0VFhamuzzvXTmjWSr1M5qZkLlB/vX/i7LcTpbn+HoLfyydm8Dn0BK7LXxOPGkHP1irupQEssFAD0Gbhe9P/3L6O8jtnl2L/BZScRr1uCGezbK5VwN0Esvvalpak/3QRduddRcqi8Zd1xjXhGB
Dn+DgYPt6jjjOUN0J9MS1btkzjxo3bTPgZL+4c8T2kknhfwpeE3LQ3TeZu5z4d0aNeaHGaQrHM7BcmKbXJfF10nw5i+jOxQfNkMPFwE+99jpvjvTMxjuCdtjpkQ2u3qhpgWgZBTkYtJjJYkWtFelUcOW6aV3xZGmLMWbQNeuBeKIB2trD0GscXoGTT1YJqgvW
eeustxcXFKTg4WPfff7/OnTtXzp3C6jZjqynnnpK27vyjRPT+39psNdlUS3qzBbUHJvhFyzWTy0LQeRJgfQn8S+Ov6evk6w+5/l/ZgqJC1PbQGgq41z6J1GtgT+OQkMJ0FgPq3nwJrjh+CSaT1U3FXRgYj+w+EmMs2wpN1ZJDhPp+49ZGhYXunrrq1KhtqoC3jm3mG9u+UKsjljqoWJ0hVO6OkRqi75nNp8u+9wv4Qi3ERy3E6BOj0v1l2rgTIcFa1uZDNH3/kfLo0xXfDaPi3H0dy8XHYZ8xbOtT4zdrebhEW+AhsBpe
p165duueee/Tggw/af8HlOENZiIuL0yc+Y0cHq3MoQ5HPxTMnCeYG72wAY7GQ9zTNrlRYbOm5/fHqDAxEFGtHz4lyjalH9jfO/2q4jSVxz8ZFq0gfK5d7ZzAQiMwvoQr/76nc9P7M8xNpnnuZ6GAQbLsrgLq3DEtYZO8Ym2HqGOf

wrHHjxmbSpElO6gg3G0lm5cqV9tfZ2dkmODjYvPTSS/Zlly9fNn5+fmbevHlO6BA3k9TUVCPJbN682RjD8Yayddt
tt5n33nuP4wxl4ty5c6ZBgwZm/fr1pnPnzmbs2LHGGP5dQ+mZOnWqadGiRZ7rOM7gDJypL4YrV67o22+/VXh4uMP
y8PBwbd261Uld4WaXlJSklJQUh+POw8NDnTt35rhDiaWlpUmSAgICJHG8oWxkZWXp448/1oULFxQWFsZxhjLx1FN
PqVevXrrvvvsclnO8oTTt379fISEhqlOnjvr376LKyslS1alWH5VWrVlVKSoqTusL
NLufYyuu4O3z4sDNawk3CGKPIyEh17NhRd9xxhySON5SuH374374QWFhYbp8+bIqVqyolStXqmnTpvZfcDnOUFo+/vh
j7d69W3Fxcbw8aSkvbtm21aNEiNWzYUCdOnCLL76o9u3b66efflI4g1MQ6kvAZrM5vDbG5G5yfVWrlVKSoqTusL
H6/vvv9fXX3+dax3HGUpPo0aNtHfv3p/ZBEmQ0ARl9KgPHiwNm/ebF/PcYb
0sP93s2bNFBYWpnr16mnhwoVq166dJI4zlC8uvy+GoKAgubC05DxVleMOpenpp5/W6tXrtXH
jRtWoUcO+nOM3V3169dX69atNXPmTLVo0UKzZZ8/mOEOp+vbbb5Wamqo6ucnV11ebNm/Xmm2/Xmm2/K1dX
VfkxxvKG0+fj4qFmzZZtq/fz/fz//rsEpCPXF4O7urjvvvFPr1693WL5s+/Xq1b9/eSV3hZlenTh0FBwc7Hd7rlzR5s2
bOe5QZMYYPfPPP6/PP/9c//nn/jX3bUbMtdOqDk/jsRzfvGCd2/tUUlH7QfgiRv9Uw3Y5SlG7AE8nYRiQyp+nnYxtFhcKIaCqwIN3y4Of6H+q9b
OCZY7ZlcwRDThjoYj3dcKe0IZJBRd0v4NLHWv49DFr/wMsj6RjdU/kxxBHhc79xj3n2zgHcpvm8WqvJD4qqnsOoWZc3d1rDcaQN8Hbd4ej3ecJ4qv8Z1wwcS4eL9mA
lpqbqxRdfVHp6ugYPHsy/a3AKQn0x/eUvf9GsvP0GpU6f0wgsv6Pjx47rjjju0du1a1a5mtwcJ27dqlrl272l9HRkZ
KkgYPHqyYmBhNmDBBly5d0qhRo3TmzBm1bdtW69at0x8AoTA09f2nI/uR81R3aPgHYhJWQ3Ka1xlFcn5aTHhFzSNHbXA
QRESEjh8/Lj8/PzVv3lyxsbG6//7JXGcpXxxvKE0HD16VAMGDNDJkydVuXJ2BYEvlO72Ye08+NZkmkfxV+Y
AAAAAAFB03FMPAAAAAIBFEeoBAAAAALAoQj0AAAAAABZFqAcAAAAAwKII9QAAAAAAWBShHgAAAAAiyLUAwAAAAB
gUYR6AAAAAAAsilAPAEApOnTokGw2m+Lj453dSqF06AAACgmAj1AACgSIYMGaJJkyY5uw1LsNqHPAA
A63F1dgMAAMA6srOzn+d7Ij9CMWqKAIXMmv1nn21n16tXObbEPbbMwsnUn9VIEAjcPbVuaLSQ5ZcSfPt+TZXkZF3vlLLtIvYxIEIi
eatGihbCo1BFT0mXXQyYyi0fYXaCE1PbWKRU6NwyEgPSt22bL9pmKPwt1EqU2c9nRX6bQuzz7ZeM5MqWPYEvIgBlAu5xpRTtsrLFw5
9+snb21sNGjTIFbb37t2rnj17qmLFiqpataoIiJ08uTJfPdrzpw5atCgQTw9PVW1alU9PDDuu/+eYbVahQQWx
btpUkHT16VP3791dAQIB8fHzUnzVwzm5+7tq5l+evn9e7n19zdzqrM/qp3ODWMpasCghTw9PVW1alU9/PDDDuu/+eYbVahQQW3
NBw4cUJcuXeTj46OwsuRH7ytr3sSnv9tJuLvdNJtsN+DF09fd2Ffq2bKl5s+fr5o1c8O8Hd5nFTdUjiz6mtO0PPn3S/0/fycq/+p12u
Nta/POYO+YsWKfH9OkrR161+bdfffd8LyLyUa2aNTVmzBhdusrhHDvBvDD4dHFaj8u7l5Vqm6Op5rl67o0r3dcX4ZVp655Fc0Qr8Kk
jSWrVqpVsNpu6dCOkiSdq0aZPatGkjHx8f+fv7q0OHHDCGLx+HI+HCHBpc48HcAPJlAABAkU2YMMHcdtJiYmxhwcMBs2bL
FvPvvuuuyYpKclIMo0bNZZr1qwxiYm2mZmzxhZbbRjZhEjtjY5uLSWlL27Z6zUw5b2nTOpZHGHhhzEzYzNjEuZZ9m6RN6h45CGa49D7cWJ2bIOsZGVWWE7
FRZmEhASze/duc//995bs3kZ1kduqlwOueb7b2xAXCY6xUm6i3W8+qZZ+Xm7r8zGNfkk/y7nmMZu
o/xfOnpMqL+KczYXy8nkX/vcdxJBecvMIqmzVynWdPxPTtYnnuYZHaXkKcycTrwBnKeYWDXdzm3Zn8ZrmzFQIdePZ8NOSqbdmLuYrXNs
dm7N+wtJDAWPFH0hHmq2WN+AncGgw3PujUdHbpYZb3p3b8DwD02gaGKj0gt2HKgYPmEWn0l3yoTBb5LfOcy9DYz7mzfdsUXPGe+4Hqzrbk3+N4+LrRAkPuRL9G0R
AvP3222b//v1m5syZpKFCiYhIcEYY8zOnTuNJPpPpl1+a48ePPm1OnTpnMzEzj5+dnnn32WXAPGgwAGZd+9eExMTYw4FfN6ZJvT2Zqz
fPlykYxAAGOMIdQDAFBE6enpxsPDw7z7+ru5/uWEexfee+++KeffjKSS7EEvLz179jLhz0Nx44dHcb
cddddZuLEifbXkS4zf/vY5Xkszf/c588bm81mPv8c2OMMZMnTzhy5+SQCLzzhy5IiRZ61ITE+11ckL98uXLja+vr0lPT8+
3z4YNG5qVpcuaa5ULqaQFsaWvSUh9m1ET/HKJ/9vhTNT37d+IHEOZ/jT0r3+1Z3WX8PSvx4TiEXj/JTtU6aDkSvBHNpR9zH0yRmaTNJM4m8Puxdy+Mwp2J+w1aTLHOgHx2QgxNJsVsvbywHWwGdiGNdTAMQhPE4/ONwQbLYaSdHE1GU0BJadXJk
YTHfzSNgDcHfrzsyH7h38HW9k+ZL/EEFkG+UtTSjGFXv8cQQqQ5P0KKq9yVCzuVMwpR4fjnKkt2G6bdXmx03Tfvnz3vnx1AQ9E6OD2XiHGxvLzvnd/76fZEY2j
1akGjAqGrAUoo6yVTlVuXvrUhMMTL/Xp4w9Xl7Lk31i9OFWc2k+f1kd1o2l1a5Yx+HIUBx1b12hdP0/2o4Q1e7T/2xEwRkQtZqkWaJ4ZEn2Tpf/hrCGbK8vZqdPK5G
8M+NRIyrzt0iM31cAUJLffk+Ht2rfW2U7ex5ey/wFrR9Q2RKx1WJK3CjPy+b3cj9jYwx5T2XgZ79R+Wz0qy/YlkkrKIW3ZPxK9J7vPqeWRhmSeaVcLxtIY23Pbg/Rgz
EfAjS7SjiSmGBrhXo1bxuEcKf9pvkvAu1zLq+g8zdbAM2BPbbpGOAQ21SRUVmL96Rz9cPpi3/xrmz0NMHXUA/EJxmv+sgcLHCMy2+tXLa3yQPM+ypJxFkOOY4A6w6mmH4ER6kB0FArSy8
5CMjfv+OnMn/5nBQ8w7ScPMDSkkDMnpHxKJBuuvEsIEyy6O7nUU6i8gSYC+/xxaqG2yLEpR/fjSROpYYi5ajRkT37unM/kK6JZbo8WSM3/hvRZtefbb8a+i/0cAm
Ycd+A6CLtUVUOBZzVsRrCjX5GRQfbRMT0+78R12LDIuFXSqkZAWqq4kX29PATYvO+5D4W3lu/mIKTUX+r+TQ9iXgcNmRmVt6H0cpm5jQkgTDX2X2lk8tvLo3OS1RDYNvIUDm9r5uUW+hqXqMzIPN0r3A
UPzXMS2GUHw2ZqtTFlb/MtH5T6Z2y7tH7nWeG46k98aq2wRp3ZtfJ5vXU+7IX4XZMzncB2PXnC8EZo3pw

QIffn+v7+/lqxYoWio6N1+fJlNWjQQEuXLtXtt9+u999/X23btlWVKlXs493d3bVu3To988wz6tmzp65evaqmTZv
q7bffliT17dtXs2fP1iuvvKIxY8aoTp06WrBggf1r3Uqifv36+vOf/6yePXvq9OnT6tmzp+bMmWNfP2bMGKWnp+u
ZZ55RamqqmjZtqtWrV6tBgwaFrtG8eXNt3rxZzz//vDp16iRjjOrVq6e//OUvhd6Gq6ur3nzzzTb3wwguaMmWKOnX
qpGXLlunnn3/WwoULderUKVWrVk2jR4/W8OHDi/QeAAAgSTZjjHF2EwAA4MbWp08fdezYURMmTHB2K4qOjtaqVas
UHx/v7FYAAHA6Lr8HAADX1bFjRw0YMEjYRw0YMMDZbQAAgD/g8nsAAHBdN8IZezYAAkBuX3wMAAAAAYFFcfg8AAAAAgEUR6gE
AAAAAsChCPQAAAAAAFkWoBwAAAADAogj1AAAAAABYFKEeAAAAAACLItQDAAAAAGBRhHoAAAAAACzq/wP9Rjm2QO0
/BQAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 1200x500 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"trials_logvar = {1: logvar(trials_csp[1]),\n",
"   2: logvar(trials_csp[2])}\n",
"plot_logvar(trials_logvar)"
]
},
{
"cell_type": "code",
"execution_count": 33,
"metadata": {},
"outputs": [
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAABKUAAAHqCAYAAADVi/1VAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAACUmUlEQVR
4nOzdd3xV9f3H8fe9NzeLJEAYYYUlU5YIDkTBRTQIttZRUy0VdRWaX9WtK1CW9GqFNsKah2Iq9S6FZU4WAIqiKO
yZYQRVhhhZNzc+/39cXMviZn35uau83o+HjySe+4Z3w/3cj7cz/0OmzHGCAAAAAAAAAAgje6QbAAAAAAAAAOuhKAU
AAAAAAICwoygFAAAAAACAsKMoBQAAAAACAsKMoBQAAAAAgLCjKAUAAAAAAICwoygFAAAAAACAsKMoBQAAAAAgLCjKAUAAAAAAIC
woygFAAAAAACAsKMohZCZO3eu+vTpo5SUFNlsNn399de99de6//77ZbPPPBbcP+ZKdvsnyWzJuzQRnRL0jUJ0WVsMsM8XYc++Crsx
j3QwAMWrLLi2y2Wz1uo8Ekk9sNpvuv/9+PZs2fLLZrNpy5YtwTUUYfXyyyyfRLaTZ84/uZjBH9gK8Ih7ZpzfmPrw82rYGjyA
///3kaYuiKIWQ2Lt3r8aPH68Tjjh/wJxPZu7ZHRxvgMjpQf0YC+Rf+yLiXZN6SCDvb4jEgEAnF5dGNV/yJc4qgBozR7MXsDDXqOnfD9rzqE21rd4BHvkb8Nu8/xsVZ
zJvh0qlBW+bRtE6zAhq3d5fn7s2LHg/1N7Ff1JzwemzE0H99i8fq9B0WhWprmsOk+0RIWtUJJsG
b6+kZ1mPyBJ4mZKKVKBC+cIHw+l9TD4yNH8tyeZw2rsCjAfrBSkrd/4BhwtD7jtGSxhzOLAFTDNJNkCOc9SbuUSN2ihgOCzpG
FhgiXqnPbu8j7XzkR94xPRlFJgpBbtZZGVYF+N6gh7E3UBRXYo8BE4iWp/SrwTHrfR0NvIrU2KTJf9yyc+Vx3fhbJ2qHStNhQkk
9GgsvGMJK8CxLfKtiqFwKjWXpC4aU33YQZv3Jzn4XuZ9uDIsk+BP+uFqQ0fBfSG8xO
/QA39zNq7xJhfGiX1OGuaq6Q7/aCVVoU4GxuKiBMkdnBYkX7Nsxh5U8NJvKQDhp5KtT4oU5wqjWUb+Lt1Vd
vuq6SjZ1kk6QYjoVGHc8FHzHqeCa2dJy8eFfm1/ss3xoHhd9d9cfRLi46tfQjRR0Gc8FXk0wk5
4rKjFsmI86xbRsFrSHlJB8k6vNYPlGSeONYM3cmFcxMBr7WcMdbkJlwxQjlp1Bz+oMN/8xnFdy1BymaQ5pWz4J8jwK+Bb
SFZ52dLf2z/LuX0cf/gCfg1q4m2R3Vl54MJhk98a1zOrEKHSdkRf2sVo/Gt5RvGaVIcEMzUgP1+pjPR1JAu9Uf1UbMVAV7Mb2
DQUK0rVwx5zHbjc4nQiQ5ufFjDDuwobXZ7bVs7F0C3fRwYU24P6UXYvpk0DBzSDyq9eZepPHIH8WVeCG+BVVsipbkuLMmNHJr
hN6uPtbR6RV9nhbA1vYCfX0ZpkY2vp7MVLNSu3TIwNOYm7+FbPQo2PJwaprO0qLpMPcS7yGAgI3oHGmUx4bP0LPrLzYVbxLn

```
njCTz+9//3ixbtswsW7bMbNu2rca2bd++3bRt29a0bNnSTJ8+3Xz00Udm7ty55sYbbzRr1qwxxhizZ88e0759e9O
qVSvzxBNPmA8++MDcdtttRpK55ZZb/OfavHmzkWQ6depkLr74YvPuu++aF1980WRlZZkePXqY8ePHmxtvvNG8//7
75oknnjBpaWnm4osvrhJ7+/btTceOHc2zzz5r5s2bZ66++mojyTz88MP+/QJtU+fOnc2FF15o3nzzTfPmm2+afv3
6mebNm5uDBw/69/3kk09MYmKiOeuss8zcuXPNBx98YK6//nojyTz33HP+/Xx/x127djW33367+fDDD83TTz9tmjd
vbs455xz/fl999ZXp2rWrGThwoP+1+Oqrr+p8vwCIH757fM+ePc2f/vQnk5uba+666y4jydx2222mV69e5u9//7v
Jzc01N9xwg5FkXnvtNf/xvntYxXvQRx99ZBwOhznzzDPN66+/bl599VVzyimnmI4dO1bKJx6Px5xzzjkmKSnJ/OU
vfzHz58839913n+natauRZO677z7/vr772ubNm/3bXnjhBWOz2cwll1xiXn/9dfPOO++Yiy66yDgcDvPRRx/VGnd
hYaHp06ePadKkiZk6dar58MMPzWuvvWZ+/etfm08++cQYYxRUZHp37+/adKkiXnkkUfM/PnzzR/+8AeTkJBgRo0
aVel8vtxyxhlnmNdff9288cYbpkePHiYzM9PceeedZsyYMebdd981L730kSnKyjL9+/c3Ho/Hf/zw4cNNixYtTLt
27czf//538+GHH5pf/epXRpK59dZb/fsF2qbOnTubU0891fznP/8x8+bNM2effbZJSEgwP/zwg3+/77//3jRt2tT
069fPzJkzx8yfP9/85je/MXa73dx///3+/T799FP/Oa+++mrz3nvvmVdeecV07NjRdO/e3ZSVlfnPN3ToUNOmTRt
/blm2bFmtrweA+FTdvXvWrFlm2rRp5u233zzYLFy40zz//vBkwYIDp2bOnKS0t9e93wQUXmFatWpmnnnrKLFiwwLz
55pvmj3/8o/n3v/9tjAnuXvOHP/zBSDKXXnqpefXVV838+fPN9OnTzR/+8Af/Pg888ICRZK666irz3nvvmTlz5tv
uvPz/dv69Oljhg8f/uO/UgAW4bvHP/roo5W2n3TSSUaSef311/3bXC6XadWqlbn00kv926rKLFiwwLz55pvj+Am
79ddfL2OMPvnkk0rbR40aJbv9+D8d37l/PFGvb3teXl6l7X369NGAAQMqbRs3bpwKCwv11VdfBdWm0aNHy+Fw+B/
3799fkrR161ZJ0saNG7V27VpdffXVPFGvb3teXl6l7X369NGAAQMqbRs3bpwKCwv11VdfBdWm0aNHy+Fw+B/
RI/3bEhIS1K1bt1rvIUePHtWXX36pSy+9VMnJyf7t6enpuvjiiyvt++mnn0qS/77mM27cuDrbu3TpUu3fv1/XXXd
dpfuhx+PRhRdeqC+//FJHjx6t8fj3339fPXr00Pnnn1/jPp988omaNGmiwmn
jXw4ZOXJkpSElvu0//ntMT0+vct8eN26cPB6PFi1aFHSb0tPT/Y+zsrLUunVr/7WLi4v18ccfa+zYsNTa2SW4q
Li6sMJSe3AGiIPXv26Oabb1Z2drYSEhLkdDrVqVMnSao0HO3UU0/V7Nmz9ec//v1nLly+Xy+Vq0HVzc3Pldrt1662
31rjPsmXLVFRU5L+n+mRnZ+vcc8+tco+12WwaNWqU/7EvT7Zt21YDBw70b8/MzKx0763oxznwjDPOUKdOnfw5Mpg
2/Tjf9u/fv9K13333XTVr1kwXX3xfv+SSedpDZt2mjBggWvNNH/+Pk5GT16NGD+z78KEohItq2bVt
l27Bhw/Tmm2+27cuDrbu3TpUu3fv1/XXXd/4X98j60pT/74vu/bXl3+rClfVrz
vN7RNSUlJla69e/duHTx4UImJiVXu/bt27arzvu87J/d9+LD6HiKipokEx4wZozFjxqXtGnTNG7cOHX
u3FlDhgwJ+DqtWrXS9u3ba92nRYsWys/Pr7LdN6l3y5YtA75ubapLhr5tvpt2qNvk23/y5Mm69mM69NJLq92nZ8+eAZ0
TAEKtefPmstlstd4nfVq0aKGysjIVFBRU+g9vfT5w+O6J//jHP2eo9PfXPL559/LmNMpZy3Z88elZWVhTy
3VFy8w6e63BLKNjVv3lwOh0Pjx4+vsfdAly5dAjonANTkf//7n7755hvnHnj1b1113nX/RkzZmjGjBn
Ky8vT22+/rvcIPeanJwcORwOzZo1qQ9hgwZopSUFL344ouuYVtm/fvl2ffPJJDXhdwCoDq+Yrrv84D
Pk08+WetxHTt21G233aYIPeanJwcORwOzZo1qQ9hgwZopSUFL344ouVtm17pW9n
Dhw/rnXfeqbSv7z740ksvVdr+8ssv13mdoUOHqlmzZlq9enWN90Rfj6TqjBw5UuvXr68yllLqi8847T0eOHNGbb75
ZafucOXP8z4fS4cOH9fbbb1vLLL8tut2vYsGGN0qbU1FSdc845WrdI+lpaWpXbt2/mFtPzZ16lS9//77lS9dPBw8e1Ac
OOzVnzhyNHj1NHj1aU6dOVadOnf1009+8hPdf//9atu2rV588UXl5ubqiQ
1SpuefPJJjRw5UhdccIGuv/56tW/fXXw34nP369dO/56tW/fXv379e/Hk
```

AwOdPf/qTLrzwQo0YMUK/+c1v5Ha79dBDD6lJkyaVhtTl5ORo2LBhuuuuu3T06FENHjxYn332mV544YU6r5GWlqZ
//OMfuu6667R//35ddtllat26tfbu3atvvvlGe/furfWDxx133KG5c+dqzJgxuvvuu3XqqaeqqKhICxcu1EUXXaR
zzjlH1157rR5//HFdd9112rJli/r166clS5bogQce0KhRo2qdjyoYLVq00C233KK8vDz16NFD8+bN07/+9S/dcss
t/rk8GqNNjz32mM4880yddZZuuWWW9S5c2cdPnxYGzdu1DvvvFNr4a4a4m/fr10+uvv65Zs2Zs2Zp0KBBBstvtGjx4cMD
nARBfevXqpRNOOEF33323jDHKzMzUO++8o9zc3Er7HTp0SOecc47GjRunXr16KT09XV9++aU++OCDSiMGArnXdO7
cWffcc4/+9Kc/qaioSFdddZWaNm2q1atXa9++qtXa9++fZoyZYqaNWumP/zhD7rnnnnt07bXX6qqrrlJBQYGmTJmi5ORk3Xf
ffSH/O1mxYoUmTJigyy+/XNu2bdOO9996r9u3b+6fnaIw2/exnP9NLL72kUaNm/kNO3i1Obd++XZ9++qn+qn
GjBmjsWPHBnTOvn37SpKeeeoppaenKzk5WV26dAnqiw3EoEjOso74Eejqe6NHj65yjnfffffeMHDnStG/f3iQmJpr
WrVubUaNGmcWLF1fa75VXXjG9evUyTqezYbTZs2bYzT6TTt2rUzV1xxhdm9e7d/n61bt5px
ZFi1aGKfTaXr27GkefvjhSqvs+VaJevjhh+sVjC+//LJK7P/9739TGJiouncubOZPn16lXY3pE3GmGr
/br755htzxRVXmNatWxun02nj33XPPE0/U2u6KcVZcFWPLli0mJyfHpKen+5c0B2AdvnvnLY22/7rr
rTJMmTarsP3z4cNOnTx//4oHHkjM3yw2nxy8OBBc+ONN5pmzZqZ1NRUM2LECLN
27do6V9/zWbhwoRk9erTJzMw0TqfTVNGlTTx9TqfH/f3owePrK/bw6BK4g/+tw8jz
gbr75ZtO2bVuTkJBgOnXqZCZPnmyKi4srnUuSufXXwYtTn+P5eFyxYYAYPHmySkpJM27ZtzT333FNlxdWGtMk
Ybx677rrrqrT1xhtvNO3btO9atXKnHHGGf7Vn2pqd8U4H74H9u/fbyp677DLrFkzYAKyhunv36tWR
rzYgRI0x6erpp3ry5ufzyyl01eXl6l+35xzbcG5+eabTf/+/U1GRoGfP/Pdd5/+PHjTbNmzfyr7Q3XK8QP1
mOTnZpKWlmYEDB1bJYU8//bSQBjTPHbgfP/hzVt2tSMGTPGDfP/PN+PHjTbNmzfyr7G3YsKHK8Q1
pU3X51+VymUceecQMGDDAJ4LGdDDA/3fRq1cv88tf/rLS9Wv63Dd8+PAqK3/8PA==

gSpErAAB1iaZcQVEKgDU5U73fLEjyeDwqPHxYGenpsoerm22Abr/9dr399ttatGiROnTo0AiNAgBUQa4AANSFXNE
gFKUAWJPNdryrq8cjOd3ex+FIHgEwxuj222/XG2+8oQULFqhLly6RbhIAWAe5AgBQF3JFg1CUAoAoduutt+rll1/
WW2+9pfT0dO3atUuS1LRpU6WkpES4dQCAaECuAADUJVpzRXSV7gAAlcyaNUuHDh3S2WefrbZt2/r/zJ07N9JNAwB
ECXIFAKAu0Zor6CkFAFHMGBPpJgAAohy5AgBQl2jNFfSUAgAAAAAAQNhRlAIAAAAAAEDYUZQCAAAAAABA2FGUAgA
AAAAAQNhRlAIAAAAAAEDYUZQCYBkejyfSTWh0VogRABqTFe6jVogRABqTFe6j4YoxISxXAYAISkxMlN1u186dO9W
qVSslJibKZrP5n/d4PCotLVVxcbHs9tis1RtjVFpaqr1798putysxMTHSTQKAmEKuAADUhVwRehSlAMQ9u92uLl2
6KD8/Xzt37qzyvDFGRUVFSklJqZRUYlFqaqo6duwYs0kQACKFXAEAqAu5IvQoSgGwhMTERHXs2FFlZWVyu92VnnO
5XFq0aJGGDRsmp9MZoRY2nMPhUEJCQswnQACIFHIFAKAu5IrQoigFwDJsNpucTmeVBOFwOFRWVqbk5OSYTh4AgIY
jVwAA6kKuCB367AIAAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAAIOwoSgEAAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAA
AIOwoSgEAAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAAIOwoSgEAAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAAIOwoSgE
AAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAAIOwoSgEAAAAAACDsKEoBAAAAAAAg7ChKAQAAAAAAIOwoSgEAAAAACD
sKEoBAAAAAAAg7ChKAQAAAAAAIOyCKkrNnDlTXbp0UXJysgYNGqTFixfXun9JSYnuvfdederUSUlJSTrhhBP07LP
PBtVgAEBsIFcAAOpCrgAAa0sI9IC5c+fqjjvu0MyZMzV06FA9+eSTGjylypFavXq2OHTtWe8wVV1yh3bt365lnnlG
3bt20Z88elZWVNbjxAIDoRK4AANSFXAEACLgoNX36dN10002aMGGCJGnGjBn68MMPNWvWLE2bNq3K/h988IEWLly
oTZs2KTMzU5LUuXPnhrUaABDVyBUAgLqQKwAAAQ3fKy0t1cqVK5WTk1Npe05OjpYuXVrtMW+//bYGDx6sv/71r2r
fvr169Oih3/72tyoqKgq+1QCAqEWuAADUhVwBAJAC7Cm1b98+ud1uZWVldelZWlXbt2VXvMpk2btGTJEiUnJ+u
NN97Qvn37NHHiRO3fv7/G8d8lJSUqKSnxPy4sLJQkuVwuuVyuQoccb72xlq7A2WVOCXrxGqVOCXrxBqu+MgVgbP
aezDe45SsE6tV4pSsEyu5InpZ7T0Y73FK1onVKnFK1ok1lPEFPHxPkmw2W6XHxpgq23w8Ho9sNpteeuklNW3aVJK
3q+5ll12mxx9/XCkpKVWOmTztmqZMmVJl+/z585WamhpMkyMuNzc30k0IC6vEKVknVqvEKcV/rMeOHQvr9cgVgYv
396CPVeKUrBOrVeKU4j9WckX0i/f3oI9V4pSsE6tV4pTiP9ZQ5oqAilItW7aUw+Go8u3Fnj17qnzL4dO2bVu1b9/
enzgkqXfv3jLGaPv27erevXuVYyZPnqxJkyb5HxcWFio6O1s5OTnKyMgIpMkR53K5lJubqxEjRsjpdEa6OY3GKnF
K1onVKnFK1om1oKAgLNchVwTOKu9Bq8QpWSdWq8QpWSdWckX0ssp70CpxStaJ1SpxStaJNZS5IqCiVGJiogYNGqT
c3FyNHTvWvz03N1djxoyp9pihQ4fq1Vdf1ZEjR5SWliZJWr9+vex2uzp06FDtMUlJSUpKSrVtjudzph9YWO57YGwSpySdWK1Spx
S/McartjIFcGL5bYHwipxStaJ1SpxSvEfK7ki+sVy2wNhlTgl68RqlTil+I81lLEFPHxv0qRJGj9+vAYPHqwhQ4b
oqaeeUl5enm6++jduzYoTlz5kiSZsyYoc6dO6tNde02vvfZayIIAAEQXcgUAoC7kCgB
AwEWpK6+8UgUFBZo6dary8/PVt29fzZs3Ky506dZIk5efnKy8vz79/aWmpfvvb32rWmpfvvb32rHjh1KSUlRnz599N5772N5772nUqFG
hiwIAEFXIFQCAupArAABBTXQ+ceJETZw4sdrnZs+eXenXXfdpbvuuiuYywAAYhi5AgBQF3IFAFibPdINAAAAAAA
AgPVQlAIAAAAAAEDYUZQCAAAAAABA2FGUAgAAAAAAEDYUZQCAAAAAABA2FGUAgAAAAAAQNhRlAIAAAAAAEDYUZQCAAAAAABA2FGUAgAAAAAAQNhRlAI
AAAAAAAEDYUZSCdXz+pPTBPZIxkW4JAAAAAACWR1EK1uBxS/N/Ly1/XDqYF+nWAAAAAABgeRSlYA2H8yV3qff30iO
RbQsAAAAAAKAoBYs4uO34766iyLUDAAAAAABIoigFqzhUsSh1LHLtAAAAAAAAkihKwSoObj3+u6s4cu0AAAAAAAC
SKErBKg7SUwoAAAAAgGhCUQrWUHHFPeaUAgAAAAAg4ihKwRqYUwoAAAAAgKhCUQrxzxjp0Pbjj+kpBQAAAABAxFG
UQvw7skcqqzC5eRlFKQAAAAAAIo2iFOJfxaF7Ej2lAAAAAACIAhSlEP8qTnIuUZQCAAAAACAKUJRC/KtSlGKicwA
AAADB2XekRNv285kCCAWKUoh/vuF7CSnen/SUAgAAABAEY4x+Omupzp++UIXFrkg3B4h5FKUQ/3w9pVp28/6kpxQ

AAACAIOQfKtbWgmMqKfNoT2FJpJsDxDyKUoh/B8t7SrXs6f3pKq55XwAAAACowdpdhf7fi13uCLYEiA8UpRDfjDk
+fK9VL+9Phu8BAAAACMKa/MP+34soSgENRlEK8a3ogFR6xPt7y+7enwzfAwAAABCE1fn0lAJCiaIU4ptvPqkmraW
U5t7f6SkFAAAAIAhrKhSlikopSgENRVEK8c03dK9ZR8npW32PnlIAAAAAAlNU6taWfUePP6anFNBgFKUQ33w9pZp
lVyhK0VMKAAAAQGDW7T4sjzn+uMTliVxjgDhBUQrxzbfyXtNsyZnq/b2M1fcAAAAABKbi0D2JnlJAKFCUQnxj+B4
AAACAEFhLUQoIOYpSiG8Ht3p/VixKecoktytybQIAAAAQc9bkH5YkpSclSGL1PSAUKEohvh2s2FMq9fh2eksBAAA
AqCdjjNbs8vaUOqljM0n0lAJCgaIU4ldxoVR80Pt702zJkSjJ5n3MZOcAAAAA6mn7gSIdLi6T02HTie0yJDHRORA
KFKUQv3zzSaU0l5LSJJvteG8pilIAAAAA6sk3yXm31unKSHZKkopK6SkFNBRFKcSvikP3fPyTnVOUAgAAAFA/vvm
kerdJV7LTIYnhe0AoUJRC/DqY5/3ZNPv4NnpKAQAAAAiQr6dU77YZSikvSjHROdBwFKUQvw6VF6Wq7SnFROcAAAA
A6sc3yXnvthlKSfR+jKanFNBwFKUQvxi+BwAAAKCBjpaUaWuB90vt3m3TlZxATykgVChKIX5VO3yPnlIAAAAA6m/
tLu98Uq3Tk9QiLUnJib6iFKvvAQ1FUQrxy7f6XrNqilJlxeFvDwAAAICYU3E+KUn+OaUYvgc0HEUpxKfSY9LRvd7
fKw3f8010Tk8pAAAAAHX7cVHKv/peKUUpoKGCKkrNnDlTXbp0UXJysgYNGqTFixfX67jPPvtMCQkJOumkk4K5LFB
/h7Z7fyamS8nNjm9nTikgbMgVAIC6kCsQC44XpdIlHe8pVVJGUQpoqICLUnPnztUdd9yhe++9V6tWrdJZZ52lkSN
HKi8vr9bjDh06pGuvvVbnnXde0I0F6s2/8l62ZLMd386cUkBYkCsAAHUhVyAWeDzGP6fUiT8evkdPKaDBAi5KTZ8
+XTfddJMmTJig3r17a8aMGcrOztasWbNqPe6Xv/ylxo0bpyFDhgTdWKDefJOcVxy6J1UYvkdPKaAxkSsAAHUhVyA
W5O0/pmOlbiUm2NWlZRNJUrLT+zG6yOWWMSaSzQNiXkBFqdLSUq1cuVI5OTmVtufk5Gjp0qU1Hvfcc8/phx9+0H3
33RdcK4FAHSyf5LziynuSlJDs/UlRCmg05AoAQF3IFYgVvqF7PbLSlODwfnz2rb7nMZLLTVEKaIiEQHbet2+f3G6
3srKyKm3PysrSrl27qj1mw4YNuvvuu7V48WIlJNTvciUlJSopKfE/Liz03ghcLpdcLlcgTY44X3tjrd2BirY4HQe
2yi7JndFengptsjuS5JDkLjlaaXsgoi3WxmKVOCXrxBqu+MgVgbPaezDe45SsE6tV4pSsEyu5InpZ7T0YLXF+v+O
gJKlnVpq/TQ7j8T9/+FixMlKcQZ072mJtLFaJU7JOrKGML6CilI+t4hw9kowxVbZJktvt1rhx4zRlyhT16NGj3ue
fNm2apkyZUmX7/PnzlZqaGniDo0Bubm6kmxAW0RLnmVu+VQtJX/2wTzv3z/NvP2H3VvWVtHPLRn01b16Nx9dHtMT
a2KwSpxT/sR47Ft651MgVgYv396CPVeKUrBOrVeKU4j9WckX0i/f3oE+0xLlwrV2SXZ792zRvnneKEGMkuxzyyKb
3PsxV08SGXSNaYm1sVolTiv9YQ5krbCaAQbClpaVKTU3Vq6++qrFjx/q3//rXv9bXX3+thQsXVtr/4MGDat68uRw
Oh3+bx+ORMUYOh0Pz58/XueeeW+U61X2jkZ2drX379ikjIyOgACPN5XIpNzdXI0aMkNMZXAU9FkRbnAl/7yfb4Xy
VXT9fpv3J/u32Fc/K8eFd8vS6WO6fPhfUuaMt1sZilTgl68RaUFCgtm3b6tChQ416LyVXBM4q70GrxClZJ1arxCl
ZJ1ZyRfSyynsw2uI859FF2n6wWC/eOFindcn0bz/pzx/raIlbH915pjplBlfgjLZYG4tV4pSsE2soc0VAPaUSExM
1aNAg5ebmVkoeubm5GjNmTJX9MzIy9N1331XaNnPmTH3yySf673//qy5dulR7naSkJCUlJVXZ7nQ6Y/aFjeW2ByI
q4iwrlQ57u30ntOwqVWxPcpokyV5WLHsD2xkVsYaBVeKU4j/WcMVGrgheLLc9EFaJU7JOrFaJU4r/WMkV0S+W2x6
IaIizsNil7QeLJUn9OmRWak+K06GjJW6VGVuD2xkNsYaDVeKU4j/WUMYW8PC9SZMmafz48Ro8eLCGDBmip556Snl
5ebr55psl5ZMnT9aOHTs84c2e129e3bt9LxrVu3VnJycpXtQMgUbpdkpIQUqUnLys85U7w/megcaFTkCgBAXcg
ViHZr8w9Lkto1TVbT1MofwpOd3l57RaXusLcLiCcBF6WuvPJKFRQUaOrUqcrPz1ffvn01b948derUSZKUn5+vvvL
8kDcUqDf/ynsdpB/PSeArSpVRlAIaE7kCAFAXcgWinW/lvd5tqw5PSikvShW7PFWeA1B/QU10PnHiRE2cOLHa52b
Pnl3rsffff7/uv//+YC4L1M/B8v+8NOtY9Tl6SgFhQ64AANSFXIFoVltRKtlflKKnFNAQ9kg3AAi5Q+U9pZplV33
OWT4JoSu8K8sAAAAAiC3rd3uH7/Vqm17lOV9PqSKKUkCDUJRC/PEP36uuKEVPKQAAAAB1O1TkkiS1aFJ1svzkRHp
KAaFAUQrxxz98r1PV5/w9pShKAQAAAKiZb76olPICVEXJCd6P0vSUAhqGohTiz7EC78+0VlWf8/eUYvgeAAAAgJr
5ekElO6t+bPYVqlh9D2gYilKIP66j3p/OJlWfS0j2/vSUSW5X+NoEAAAAIKb4ekH55o+qyLetpIzV94CGoCiF+FN
a3gsqMbXqc84K2xjCBwAAAKAaxphai1K+1ffoKQU0DEUpxB/f0DxnNUWphCRJtvL9KEoBAAAAqKrU7ZEx3t+Tait
KMacU0CAUpRBfPJ7jRanEaobv2WwVJjtnXikAAAAAVRWXHh+WV9vwPVbfAxqGohTiS1mF3k/V9ZSSKkx2Tk8pAAA
AAFX5ekA57DY5HbYqz/smP6enFNAwFKUQX0or9H6qsSjl6ylFUQoAAABAVf6V9xLsstmqFqV8q+/RUwpoGIpSiC+
+lfcSUiR7DW9vZ/kKfGUUpQAAAABU5Z/kPLHq0D3p+JxSxS5W3wMagqIU4kttK+/5MHwPAAAAQC18RankauaTqri
d1feAhqEohfjiX3mvmknOfZjoHAAAAEAtiusoSqWw+h4QEhSlEF9c9JQCAAAA0DC+olR1K+9V3M6cUkDDUJRCfPE
N36tpkvOKz9FTCgAAAEA1ikq9c0XVVJTyrb5HUQpoGIpSiC++ic4Taxu+R08pAAAAADXzD9+rY6Jzhu8BDUNRCvH
F31MqpeZ9EspX36MoBQAAAKAa/onOE6r/yOxblY/V94CGoSiF+OIKZPgeRSkAAAAAVfnnlKKnFNCoKEohvpQyfA8
AAABAw9R3ovPSMo/cHhO2dgHxhqIU4ktAPaWY6BwAAABAVf7he3UUpSSppIzeUkCwKEohvvjmlEqsrShFTykAAAA
ANfOtvldTUSqpwlxTRaUUpYBgUZRCfPGtvuesx/C9MopSAAAAAKoqLqt9+J7dbvMXpphXCggeRSnEF3pKAQAAAGi
g4lLf8L2aPzKzAh/QcBSlEF/qNacURSkAAAAANSuqY/U9SUpO8BWl6CkFBIuiFOJLvVbfY6JzAAAAADUrrmOic+l
4wYrhe0DwKEohvtBTCgAAAEAD1bX6XsXn6CkFBI+iFOJLveaUoqcUAAAAgJoVlc8TVdNE59Lx+aZYfQ8IHkUpxJd
AVt9zFTd+ewAAAADEnBJX7avvVXyO4XtA8ChKIb7Up6dUAsP3AAAAANTs+PC9WlbfKy9KlbD6HhA0ilKIL/45per
TU4rhewAAAACq8g3Jq8+cUvSUAoJHUQrxw5gKq+/VY6Jzj0tyuxq/XQAAAABiim/yct8Ke9WhKAU0HEUpxA+3SzL
lCaHW1fcqPMcQPgAAAAA/Ulw+JK+2nlIpifbyfSlKAcGiKIX44ZvkXJISaxm+l5AkyVZ+DEUpAAAAAMe5PUal7nq
svpdATymgoShKIX74Jjm3OyWHs+b9bLbjvaXKKEoBAAAAOK5iz6daV98rH9pXXEpRCggWRSnED1c9Vt7zcSaXH0N
RCgAAAMBxFXs+JSXU/JHZN7SvmNX3gKBRlEL88E1yXtvKez6+nlKswAcAAACgAt/Ke0kJdtntthr3Y6JzoOEoSiF
++ApMvtX1auPbh55SAAAAACooKat75T3p+NA+ilJA8ChKIX6UBjJ8j6IUAAAAgKqKSstX3kuooyjF6ntAg1GUQvx
wBTN8j6IUAAAAgON8PZ/q6inlK1pRlAKCR1EK8YOeUgAAAAAayFdkSq5l5T1JSk5k+B7QUBSlED/8PaXqUZRK8BW
lmOgcAAAAwHH+nlLO2j8up7D6HtBgFKUQP/w9peozfI+eUgAAAACqqndPKd9E56X0lAKCRVEK8cO/+l4gw/foKQU

AAADgOF+RKaWOotTxnlIUpYBgUZRC/CgtH75Xr55STHQOAAAAoCp/T6k6JjqnKAU0XFBFqZkzZ6pLly5KTk7WoEG
DtHjx4hr3XbJkiYYOHaoWLVooJSVFvXr10t/+9regGwzUKJieUmXFjdcewOLIFQCAupArEI2KyueI8q2uV5Pk8jm
nilxuGWMavV1APEoI9IC5c+fqjjvu0MyZMzV06FA9+eSTGjlypFavXq2OHTtW2b9Jkya67bbb1L9/fzVp0kRLliz
RL3/5SzVp0kS/+MUvQhIEICnA1fd8PaUYvgc0BnIFAKAu5ApEK/9E54m19+Hw9aTyGKnU7VFSHUUsAFUF3FNq+vT
puummmzRhwgT17t1bM2bMUHZ2tmbNmlXt/gMHDtRVV12lPn36qHPnzrrmmmt0wQUX1PotCBAU/+p79Rm+l1x+DMP
3gMZArgAA1IVcgWhV4gpsTimJFfiAYAXUU6q0tFQrV67U3XffXWl7Tk6Oli5dWq9zrFq1SkuXLtWf//znGvcpKSl
RSUmJ/3FhYaEkyeVyyeVyBdLkiPO1N9baHahoiNNRclR2SWWOJJk62mG3J8khyVNyVO4A2xwNsYaDVeKUrBNruOI
jVwTOau/BeI9Tsk6sVolTsk6s5IroZbX3YCTjPFrivbbTbquzHQ67TW6P0eFjxUoNcBxSNMQaDlaJU7JOrKGML6B
/Nvv27ZPb7VZWVlal7VlZWdq1a1etx3bo0EF79+5VWVmZ7r//fk2YMKHGfadNm6YpU6ZU2T5//nylptZjaFYUys3
NjXQTwiKScQ7dvU0tJa363zrt3DGv1n07FmzQQEl7dm7V5/Nq37cmvKbxJ95jPXYsPMNVyRXBi/f3oI9V4pSsE6t
V4pTiP1ZyRfSL9/egTyTj3LDJLsmuvM0bNG/e+lr3TbA55JZNH+R+olYpwV2P1zT+xHusocwVAc8pJUk2m63SY2N
MlW0/tnjxYh05ckTLly/X3XffrW7duuqq66qdt/Jkydr0qRJ/seFhYXKzs5WTk6OMjIygmlyxLhcLuXm5mrEiBF
yOp2Rbk6jiYY4Hc88Kh2RBp46VCd1z6l1X9vqEinvabVunqZRo0YFdJ1oiDUcrBKnZJ1YCwoKwno9ckX9WeU9aJU
4JevEapU4JevESq6IXlZ5D0ZDnPP/8620d5cG9D1Ro4Z0qnXfP323QCVHSnX60LPUs016QNeJhljDwSpxStaJNZS
5IqCiVMuWLeVwOKp8e7Fnz54q33L8WJcuXSRJ/fr10+7du3X//ffXmDySkpKUlJRUZbvT6YzZFzaW2x6IiMZZvpJ
eQkqGVFcbkr0Jw15WInuQ7eU1jT/xHmu4YiNXBC+W2x4Iq8QpWSdWq8QpxX+s5IroF8ttD0Qk4ywp866k1yQpsc4
2JJfPK+UytqDby2saf+I91lDGFtBE54mJiRo0aFCVrmi5ubk644wz6n0eY0ylsd1ASLgCWX2vvG8tE50DIUeuAAD
UhVyBaFZcz9X3pOOTnTPRORCcgIfvTZo0SePHj9fgwYM1ZMgQPfXUU8rLy9PNN98sydtFdseOHZozZ44k6fHHH1f
Hjh3Vq1cvSdKSJUv0yOP6Pbbbw9hGICk0gBW30vwFaXCM28CYDXkCgBAXcgViFbF9Vx9T5JSEn1FKXejtgmIVwE
Xpa688koVFBRo6tSpys/PV9++fTVv3jx16uQda5ufn6+8vDz/h6PR5MnT9bmzZuVkJCgE044QQ8++KB++ctfhi4
KQKKnFBBFyBUAgLqQKxCtisoLTEn1KEolJzgqHQMgMEFNdD5x4kRNnDix2udmz55d6fHtt9/OtxdofB63f06pevW
UcpYXrihKAY2GXAEAqAu5AtGoKICeUsn0lAIaJKA5pYCoVXEYnrMea7H69imjKAUAAADguJLy+aHqNXzP6f1ITU8
pIDgUpRAfSn1FKVtgRSl3qeQua7RmAQAAAIgtvgJTcn16SpXvU1RKUQoIBkUpxAeXb5LzVMlmq3t/Z4V5p+gtBQA
AAKCcr8BUv55S3n1Kylh9DwgGRSnEh9IAJjmXpIQkSeXFK+aVAgAAACDJGKPisvKeUol1f1ympxTQMBSlEB98c0o
561mUslUY5ldxPioAAAAAllVS5pEx3t8DGr7HnFJAUChKIT6Ulg/fS6zHyns+/qIUPaUAAAAAVF5FL5Dhe6y+BwS
HohTiQ6A9pSruS1EKAAAAgKTi8pX3Euw2OR11f1xOSWT1PaAhKEohPgQ6p5RETykAAAAAlQSy8l7F/egpBQSHohT
ig3/1PYbvAQAAAAiOb8LywItSrL4HBIOiFOJDMD2lEpjoHAAAAMBxvpX3Uuqx8p50fE4pVt8DgkNRCvHB31OK4Xs
AAAAAglPs6ymVEFhPKeaUAoJDUQrxwd9TKpDhe76JzukpBQAAAOB4cSklsX5FKVbfAxqGohTiQ1Cr75X3lCorDn1
7AAAAAMQc39xQ9Z1TyjfMj6IUEByKUogProasvkdPKQAAAAAVekrVsyiVlMDwPaAhKEohPviG7wW0+p5v+B5zSgE
AAAA4XlxKdtZzovNEVt8DGoKiFOJDUD2lksuPpSgFAAAAQCoJsKdUChOdAw1CUQrxodS3+h4TnQMAAAAITlFpYBO
d++aeKi3zyO0xjdYuIF5RlEJ8aNCcUvSUAgAAAHC8x5Nvrqi6VOxRVVJGbykgUBSlEB9Kg1l9jzmlAAAAABznmxu
qvj2lkhKOf6T29bICUH8UpRAfXL7he/SUAgAAABCcQFffs9tt/sIU80oBgaMohfhQ2pDhe8wpBQAAAEAqDnD1PYk
V+ICGoCiF+OAKYvheAj2lAAAAABxXHGBPKUlKTvAVpegpBQSKohRinzHHV99LDGT1PYpSAAAAAI4r8veUqn9Rytd
TiuF7QOAoSiH2lRVLKl9+NZiJzssoSgEAAAA4Pll5IEUp3770lAICR1EKsa+0wpxQ9JQCAAAAEKTisvLV9wIqSpV
PdM7qe0DAKEoh9vlW3nMkSfb6Jw9/TykmOgcAAAAgqbi8sOQbklcfvgIWw/eAwFGUQuwLZuU9iZ5SAAAAACopCmb
1vfKiVAmr7wEBoyiF2OfrKeUMYOiedLwo5S6V3GWhbRMAAACAmBPMROfJ9JQCgkZRCrGvoT2lJCY7BwAAAOCfrDy
wOaUoSgHBoiiF2OebEyqQlfckKSG5wjmKQ9ceAAAAADGpOIieUimJ9krHAqg/ilKIfaXlw/cCWXlPkmw2JjsHAAA
AIEkqc3vkchtJAfaUSqCnFBAsilKIfcH2lJKY7BwAAACAJKm47PhE5QGtvle+r2/lPgD1R1EKsc9XUAp0TimJnlI
AAAAAJElFFYpKSQn1/6jsG+pXzOp7QMAoSiH2lQa5+p50fF4pekoBAAAAlnZ8Pim7bDZbvY9jonMgeBSlEPtcQa6
+JzF8DwAAAICk4Fbeq7g/RSkgcBSlEPv8PaUaMHyvjKIUAAAAYGVFQay8J7H6HtAQFKUQ+/w9pYIYvkdPKQAAAAA
6PqdUoD2lfKvvUZQCAkdRCrGvtCGr7zHROQAAAIDjq+8F2lMqOZHhe0CwKEoh9rl8w/dSAj/WyUTnAAAAAI73lEp
2BvYxOYXV94CgUZRC7CsNxfA9ekoBAAAAVuaf6DwxwJ5SvonOS+kpBQSKohRinysUw/eKQ9ceAAAAADGnoavvMac
UEDiKUoh9vtX36CkFAAAAIEhBr75HUQoIGkUpxL6Q9JRiTikAAADAyoItSvnmoCpyuWWMCXm7gHgWVFFq5syZ6tK
li5KTkzVo0CAtXry4xn1ff/11jRgxQq1atWrVJGRoaGDBmiDz/8MOgGA1X455QKpijl6ylFUQoINXIFAKAu5ApEE99
E5YEO3/OtvucxUqmbyc6BQARclJo7d67uuOMO3XvvvVq1apXXvvVq1apXOOussjRw5Unl5edXuv2jRIo8YMULz
55+jiiy/WqlWrGtx4QFKF1feCGL6XwPA9oDGQKwAAdSFXINoEO9F5xSIWK/ABgQm4KDV9+nTddNNNmjdNNNmjBhgnr37q0
ZM2YoOztbs2bNmjBhgnR67677HHHlD37t31zjvvNLjxgCR6SgFRiFwBAKgLuQLRxrd6XnJCYB+
TnQ67HHabJOaVAgKVEMjOpaWlWrlype6+++5K23NycrR06dcjcPj8ejw4cPKzMyscZ+SkhKVlJT4HxcWFkqSXC6
XXC5XIE2OOF97Y63dgYpYnG6XnJ7ya9sSpQCvb7MnKkGSx3VM7noey2saf6wSa7jiI1cEzmrvwXiPU7JOrFaJU7J
OrOSK6GW192Ak4jxW6r1mosMW8PWTnXYdLXHr8LESZabUr6cVr2n8sUqsoYwvoKLUvn375Ha7lZWVVWl7VlaWdu3
aVa9zPProozp69KiuuOKKiuuOKKGveZNm2apkyZUmX7/PnzlZoaRG+YKJCbmxvpJoRFuONMKDuq0eW/f/DxInnszoCOb3N
otU6TdHDPTi2eNy+gY3lN40+8x3rsWHiGqZIrghfv70Efq8QpWSdWq8QpxX+s5IroF+/vQZ9IxLlpq12SXZs2rNW
8I2sCOtbucUiyKfeTBWoX4KwivKbxJ95jDWWuCKgo5WOz2So9NsZU2VadV155Rffff7eeusuSb9Jk+erEJ
TJvkfFxYWKjs7Wzk5OcrIyIyRjcrmUm5urESNGyOkMrGASSyIWZ2G+9J1kbA5dOPonUj3ehxXZNjeRNs1Q87R

kjRo1ql7H8JrGH6vEWlBQENbrkSvqzyrvQavEKVknVqvEKVknVnJF9LLKezCScb61f5VUsFcnD+inUYM7BHTsw2s
W6fDBYg0+/QydlN2sXsfwmsYfq8QaylwRUFGqZcuWcjgcVb692LNnT5VvOX5s7ty5uummmm/Tqq6/q/PPPr3XfpKQ
kJSUlVdnudDpj9oWN5bYHIuxxmlJJki2xiZyJiYEfn5zuPb6sKOB285rGn3iPNVyxkSuCF8ttD4RV4pSsE6tV4pT
iP1ZyRfSL5bYHIhJxlrqNJCktOTHga6ckej9au4yNzxU1sEqcUvzHGsrYAprBLTExUYMGDarSFS03N1dnnHFGjce
98soruv766/Xyyy9r9OjRNe4HBMy3ap4zyO7XzuTy8zDRORAq5AoAQF3IFYhGReWTlCc7A1t9Tzq+Yl8Jq+8BAQl
4+N6kSZM0fvx4DR48WEOGDNFTTz2lvLw83XzzzZK8XWR37NihOXPmSPImjmuvvVaPPfaYTj/9dP+3ISkpKWratGk
IQ4EluRqw8p50vJjlCs/8CYBVkCsAAHUhVyDa+Fffcwa8SL2SE7xFqSJW3wMCEnBR6sorr1RBQYGmTp2q/Px89e3
bV/PmzVOnTp0kSfn5+crLy/Pv/+STT6qsrEy33nqrbr31Vv/26667TrNnz254BLC20qPen84AZxP0caZ4f9JTCgg
pcgUAoC7kCkSb4jJvQSkliJ5SyeU9pXyFLQD1E9RE5xMnTtTEiROrfe7HCWHBggXBXAKon4b2lEoqn+DSXeotTPm
KVAAajFwBAKgLuQLRpLi8oOQbiheIlPLeVb7CFoD6CbxfIhBNShs4p1RSumQrTzpFB0PSJAAAAACxpyFzSvmOoac
UEBiKUohtrvLhe4lBDt+z2aTk8jkIig+GpEkAAAAAYk9x+STlwQzf8x1TzJxSQEAoSiG2NbSnlCSlNPP+pKcUAAA
AYEnGmJD0lCpm9T0gIBSlENt8c0o1ZC6o5Gben/SUAgAAACyppOx4MSmo1fecrL4HBIOiFGJbaQOH70n0lAIAAAA
sruKwu2B6SqVQlAKCQlEKsc0VguF79JQCAAAALM1XTHI6bHI6Av+YnJJYvvoeRSkgIBSlENt8c0olMqcUAAAAgOD
4Vs1LTgi8l5RUcU4pilJAIChKIbb5Vt9zNmD4Hj2lAAAAAEvzTVCenNiwopSvuAWgfihKIbbRUwoAAABAA/mG76U
EMZ9UxeNYfQ8IDEUpxDbmlAIAAADQQL5hd8GsvOc9jonOgWBQlEJsY/U9AAAAAA3kG3bX8J5SFKWAQFCUQmyjpxQ
AAACABiou8/WUCrIoxep7QFAoSiG2+eeUoqcUAAAAgOD4V98LsiiVlMDwPSAYFKUQ2/yr79FTCgAAAEBwihs60Xk
iE50DwaAohdgWytX3yoolV3GDmwQAAAAgtviKSb7iUqB8xayiUreMMSFrFxDvKEohdnk8UlmR93dnA4bvJaZLtvJ
/CvSWAgAAACynyNWw4XtNU5ySpFK3R8dKGcIH1BdFKcQuX0FKalhPKbtdSm7q/Z15pQAAAADLOV6UCu4jcmqiQ0k
J3mP3Hy0NWbuAeEdRCrHLN3RPkhJSGnYu5pUCAAAALKuhc0rZbDa1aJIoSSqgKAXUG0UpxK6Kk5zbG/hWZgU+AAA
AwLIaWpSSpMw0b1Fq/9GSkLQJsAKKUohdvp5SDVl5z8c3fI+eUgAAAIDlFJU2bE4pSWrRJEmSVHCEnlJAfVGUQux
yhWDlPR/f8D16SgEAAACW41t9LznI1fck+YfvMacUUH8UpRC7Sn3D9xqw8p6Pb/he8aGGnwsAAABATCkKxfA9ilJ
AwChKIXY1Rk8phu8BAAAAltPQ1fek43NKMdE5UH8UpRC7SitMdN5QTHQOAAAAWFZJCHpKMXwPCBxFKcQuVygnOm/
m/UlPKQAAAMByQjN8zzfROavvAfVFUQqxqzSEw/foKQUAAABYlq8olRSCOaUYvgfUH0UpxC5XCCc6p6cUAAAAYFm
+1fcYvgeEF0UpxC56SgEAAAAIgeLS8uF7iQ3oKVU+0fmxUreKy3teAagdRSnELuaUAgAAABACoVh9Lz0pQU6HTRJ
D+ID6oiiF2OVbfS8xBMP3fD2lXMekMhIIAAAAYBUut0dlHiOpYcP3bDabf16p/Uf4TAHUB0UpxC5fUSoUPaWSmkr
yfqtBbykAAADAOioOtUtuQFFKqrAC31FW4APqg6IUYtfBPO/PjHYNP5fdLiVneH9nXikAAADAMnxD92w2KSmhYR+
RW6Yx2TkQCIpSiE3GSPvWe39v2SM052ReKQAAAMBySspX3ktOcMhmszXoXJmswAcEhKIUYtOxgvLikU1qcUJozsk
KfAAAAIDl+HpKNWTlPR9fUYqJzoH6oSiF2OTrJdUsW3KmhOac9JQCAAAALKeotHzlvQYO3ZOkFkx0DgSEohRiU6i
H7kn0lAIAAAAsyDfReXJIekox0TkQCIpSiE37Nnh/tugeunPSUwoAAACwHP/wvQauvCcxfA8IFEUpxCZfUaplCIt
S9JQCAAAALMffUyoERakWrL4HBISiFGJTYwzfo6cUAAAAYDnF5avvhbKnFHNKAfVDUQqxp6xEOrjV+zs9pQAAAAA
0QFEoe0qVF6UOl5SppMzd4PMB8Y6iFGLP/k2S8UhJGVJaVujOS08pAAAAwHL8q+85G/7xOCPZKYfdJkk6cNTV4PM
B8Y6iFGKPf+hed8lmC9156SkFAAAAWE5xWegmOrfbbWqe6pvsnBX4gLpQlELsaYyV9yR6SgEAAAAWVFzeUyolseF
FKUlqyWTnQL0FVZSaOXOmunTpouTkZA0aNEiLFy+ucd/8/HyNGzdOPXv2lN1u1x133BFsWwGvxlh5T6KnFBBi5Ao
AQF3IFYgGvjmlQtFTSqow2TlFKaBOARel5s6dqzvuuEP33nuvVq1apbPOOksjR45UXl5etfuXlJSoVatWuvfeezv
gwIAGNxholJX3pOM9pVxHJTfjv4GGIFcAAOpCrkC08K2+lxTiolQBK/ABdQq4KDV9+nTddNNNmjBhgnr37q0ZM2Y
oOztbs2bNqnb/zp0767HHHtO1116rpk2bNrjBsDhjpIKN3t9D3VMqucL7k95SQIOQKwAAdSFXIFqEuqdUC3pKAfU
WUFGqtLRUK1euVE5OTqXtOTk5Wrp0aUgbBlTryG6ppFCy2aXMrqE9t90hJZX/B4d5pYCgkSsAAHUhVyCa7C4sliS
lJyeE5HyZTZIkMdE5UB8B/avbt2+f3G63srKyKm3PysrSrl27QtaokpISlZQc/wdcWFgoSXK5XHK5YmtYla+9sdb
uQIUrTtuu1UqQZJp1UpmxSyG+XkJyU9lKDqnsyD6Zpp2r3YfXNP5YJdZwxUeuCJzV3oPxHqdknVitEqdknVjJFdH
Lau/BcMVZ7HLryy37JUkDO6SH5LpNU7w9rvYdLqn1fLym8ccqsYYyvqBKwTabrdJjY0yVbQ0xbdo0TZkypcr2+fP
nKzU1NWTXCafc3NxINyEsGjvOzvs+0QBJu90Z+nzevJCff3ipTc0kfbkoV3ua7ql1X17T+BPvsR47diys1yNXBC7
e34M+VolTsk6sVolTiv9YyRXRL97fgz7hinPtQZuKXQ41TTTasGKxNobg7be1wCbJoY3bd2tePT6z8JrGn3iPNZS
5IqCiVMuWLeVwOKp8e7Fnz54q33I0xOTJkzVp0iT/48LCQmVnZysnJ0cZGRkaNEiLFy+ucd/8/HyNGzdOPXv2lN1u
JV5z2+Z9J26RWvYdq1PmjQn5+x4F/SVu26pR+3WX6Vn9+XtP4Y5VYCwoKwnIdckXgrPIejEScH6/Zo9nLtuqW4V1
1xgktwnJNidc0HlklVnJF9LLKezDccX79/jpJW5XTr4NGj+4TknO22Lxfz61fISU20ahRZ9a4H69p/LFKrKHMFQE
VpRITEzVo0CDl5uZq7Nix/u25ubkaNGiQ6zIt6vxfl7T+BPvsYYrNnJF8GK57YEIV5y7DhVr+mufaEDZt7p1zhBNGjVQNwztHNJeGHXhNY0/8R4ruSL6xXLbAxG
uOJds9H64PrtnVsiul9XU2wtv/zFXvc7Jaxp/4j3WUMYW8PC9SZMmafz48Ro8eLeLCGDBmip556nSnl5ebr5psleb+
N2LFjh+bMmeM/5uuvv5YkHTlyRTlyRHv37tXXX3+dXXX3+txMREnXjiiaGJAtZRsMH7M9Qr7/mkNPP+ZPU9oEHIFYg04y7Twuf
v15t6VqnOEt3o+UA/f2+SVuefrD9f0lfJIVphCUDwyBWItB0Hi7RhzxHZbdKZ3VqG7LyZ5avvHSpyyeX2yOkIeNF
7wDICLkpdeeWVKigo0NSpU5Wfn6+6+6+++tTqZ3rx56tSpkyqYqPz9feXl5Y4ZOHCg/eTJk/bNzc3+utss0YE8xlgMr2/yDJeIfYpTbSvCDJTb0/6SkFNBi5Io58/6b06nVS2wH
rVet08g1Qk6AmOp84caImTpxY7XOzZ8+uss+uss0YE8xlgMr2/yDJeIfYpTbSvCDJTb0/6SkFNBi5Io58/6b06nVS2wH

SRX+T2g+SJLk9Rj/sPaI2TZOVkdwIXdR3fi2tflPa8JG0+7vKzx3eKf3wsTcfDLjKW6Bq3kVa/KjM4kfV3ONSoUn
R8m53KucnV0tzxytz51d6IXGa/rLzal3892N68trBGtixeejbDaDeyBWIpIXrvEWp4T1ahfS8DrtNzVMTtf9oqfY
fpSgF1CaoohQQEft8Q/d6SI01H4hv+B49pQDguPUfen/mfyP96zzp1J9L5/5ef5q/XbOXbpEktWuarB5t0tUjy/u
nX/um6tkmPfhr/u916b83SvJ9ALVJ7QZK3UdIHU+XtiyRVr0kHdklLfun909qC+lYgWySct2D9HTGrZpz1SVSgkO
64X3p3TuU8M0rus/5gvqUbNX4pyZo9oSzNLhzZvDtBADEJJfbo8827pMkDQtxUUqSWjQpL0odKQ35uYF4QlEKscN
XlGrRSPNJSceH79FTCgCO2/6F92f7wdKOFdIXT8n1vze179DPJJ0myaadh4q181CxFpR/6yxJZ/dspbsu6KUT2wW
4wlXecumNmyUZqdsIqf8V0gnnSk0qzPdxwrnS2fd4h/R9NUda/4F0rECupEzdcfgavec5Ta9efoaSEsqHGzqTpUt
mSW36y8y/V5c5FqmbZ4d+/+JdevbXl/AtNgBYzNfbDupwSZmapzrVr33TkJ/fN69UwVGKUkBtKEohduxb7/3ZWJO
cS/SUAoAfO7ZfKvCufKqrX5V2fSvz7iQ59/+gfzr/rlubnar2N8zW+iMpWrf7sNbvOqy1uw5r5dYDWrBurxau36t
LTmqvSSN6qE169UP8fMNxbDabtG+j9MrPJHeJ1Osi6Yo5Nc9j5UiQel7o/XN4l0o3LtRlucn61pOga07vqFN+3AP
KZpOGTJStdS+ZV2/QScU/aI7rN3ry2T36v9tuUwIT0QKAZfiG7p3VvZUc9tCPwmiR5i1K7acoBdSKohRiR0GF4Xu
NJbl8bhF6SgGA1/YvvT9bdJdSM6WuZ+vlQf/W3venaWLC2+p99AvpjfEafP17lYbBbdl3VI/MX6d3v83XG6t26L1
v83XVqR3U+qiUu3qPNu8v0obdh7VhzxH9sPeIOmU20UvjTlDLf/9UKjrgnbfq0n/Vf2L19Db62+4B+nb/D8rKSNJ
dF/aqed8TzpXtFwtU/Mq1ytz7rSYf+IM+e3qjhk6Y7i10AQDi3qIN3qJUYwzdkyr0lDpS0ijnB+IFXwkiNng8Fea
UoqcUAITNtvKhe9mnSpL2Hi7Rg7mbNaPsMr0/dK538YkdK6W3b/cuM1Suc8sm+ue4k/XObWdqaLcWKnV79PyyPD3
8bYImvvK1Hv5wnd78eqe+31moYpdHW3YXaO9TY6UDW6RmnaSr/i0lpta7mf/bcUhPLdokSfrTmL51T7ye2UXJv8j
V1q7jJElD859XwRMjvav8AQDi2r4jJfp2+yFJ0rDuLevYOziZTZIkMXwPqAtfByI2HN4puY5J9gSpeefGu45vTqn
SI5LbJTkaYTUpAIglvvmkOpwiSZr2/hodLi5T3/YZuui8M6Vuc6QXxkrfvSq17i2d9ZtKh/fr0FQvTThdizfs1cM
frtW6nYfUrU2GemRlqFvrNHVvnaaMJLuOvXSNervX6YgtTc4r5yoprXW9m3i42KXbX1klt8doZN82yunTpn4HOpP
V6dpZevOF3jp/41/UYu8XKpt5phIuf0bqOrze1wcAxJYlG7wTnJ/YNkOtMxpnTsEWTRi+B9QHRSnEBl8vqcyu9So
UlZZ5lJgQREfA5AqTHBYfqjypLgBYjcct7fjK+3uHU7R8U4Fe/2qHbDZvbySH3SZ1GSaN/Kv03iTp46lSq15Sr9F
VTnVW91Y6vXMzzZs3T6NGDZHTWX4vN0b68B7JfK5Sk6AbS+5UxvyjeuIaT73meDLG6HevfavN+46qXdNk/WVsv4D
DvGjcbfq/J9voF7umqnfRNpmXLpNt4nKpxQkBnwsAEP0Wrm/coXsSE50D9cXwPcSGAFbeu/u1b9XzD+/r4n8s0V8
/WKulP+xTSZm7ftdxJEiJ5UuYM68UAKvbs9rbczQxXa4WPfXHt/4nSbrq1I4a2LH58f1OuUk65efe31/7ubTrf/U
7v8ctzfuttHymJClv2CP6xtFHH63Zrd+99p08HlPHCaTZS7do3ne75HTY9M+rT/Z/CAhEgsOue8aP0S+THtIy94m
yuUtlFk8P+DwAgOjn8RgtLp9PangwRamDedLiR6XPn6w0bP3H6CkF1A9FKcSGeq68V+xy641VO2SM9N2OQ5q54Ae
N+9fnOmlKrq5/7gu9/tX2uq/FvFIA4OWbT6r9yXpuWZ7W7z6izCaJuuuCnlX3vXCa1GW45DoqvXKVdGRv7ecuK5V
emyB9+bQkmzT6UXU77wb9c9zJcthteu2r7Xpg3hr/ynzV+SrvgP7y3hpJ0j2jeuvkioWyALVKT9LfrhmiRz0/kyS
Zb/4tHdga9PkAANFpdX6h9h0pVZNEhwZ1qmfeKDksrXpJmn2RNKOft2fw+3dJW5bUeEgmq+8B9UJRCrGhnivvfbX
1gErKPGqVnqTpVwzQ2IHt1TItSUUutxas26tJ//lGP+w9Uvu1fPNK0VMKgNWVr7x3uNXJmvGR9z48eWQvNUutpje
Swyld8byUeYJ0KE+ae03NRZ3SI9LLV0jfvy7ZndJlz0inTJAkjTgxS3/9aX9J0tNLNmvqu6urXblo/9FS3fbSVyr
zGI3u11bXn9G5weEO6pSp04ZdoMXuvrKbMrkX/63B5wQARBff0L0zurWse7qPgh+k138pPdJDemuitGWxJJuUluV
9fuVzNR7q67l74Fip3PXo+QtYFUUpxIZ6rry3ZKN30sIzu7XUpSd30N+uPElf3nue3v/1WerX3jtf1OL1dXx7T08
pAPAq7yn14aFsHSt1a1Cn5vrpyR1q3j+luXfVvKSm0rbl0mMDpBculb5/09szSpKz7LAcL10qbfpUcjaRxs2V+v6
00ml+OqiDfj+6tyTpuc+26IwHP9E9b3ynTeVfKng8RnfM/Vo7DxWrS8smevCn/WSz2UIS8sSzu+nFxCu8D1a9KBX
uDMl5AQDRod7zSbnLpBcvlb79t3fBpcwTpHN/L93xnTTuP959Vr9dY8/g5uVf4BgjHTxGbymgJhSlEP1KDkuFO7y
/t+hW666fVShK+dhsNvVum6FR/dpKkpZsLKj9er7JzosOBNdeAIgHRwuk/T9Ikl7Z6f1GePzpnWS311H8adVDGv+
GdyifjPTDx9Kr10l/O1H2j+/TWRv+IvvOr7wFrOvelrqdV+1pJpzVVU9cc7L6d2iqkjKPXv48T+dNX6ifz1mhP7z
1Py1av1fJTrtmXXOy0pNDt1Jqk6QE5Yz6qT739JLDuHRsAXNLAUC8KCx26aut3v/jD+9eR1Hqf69JB7ZIqS2lm3K
l21dKw/5PapYttTtJaney5HFJX79U7eFOh11NU7z5iSF8QM0oSiH6FWz0/mzSSkrNrHG3Q8dc+nbHIUnS0G5VV80
7q7t32/JNBXK5PTVfj55SAOAfuudqdoJW7rXJYbfp7J71nBC2wyBvwelXq6QzJ3mHORzdK8fyx5VevFMmvZ1044d
Sh8G1nubCvm311q1DNfcXp+v83q1ljJS7erde+jxPkncFwF5tMhoUZnXGDmyv95pfI0lyrnpeOrIn5NcAAITf0o0
FKvMYdW3ZRB1bpNa8o8cjLSn/UmLIRCn7VOnHPXIH3+D9uXK2d/9qtEhjBT6gLhSlEP32lRel6lh5b9mmfTJG6tY
6TW2aJld5/sS2GWqe6tSRkjJ9u/1gzSdiTikAkLZ7h+5tTukjSTqlc/Pq55KqTWZX6fz7pDu/l658SZ5uOdqb1lt
l182TWlUzWXo1bDabTuvaQk9fd4o+mjRcV53aUU0SHbpxaBddPjg7sPbUk91u05ixV2uVp5ucplR75z/SKNcBAIT
X2994R18Mr+tLlrXvSnvXeoejl895WEXfn0pJGdKBzdLmBdXuwgp8QN0oSiH67frG+zOA+aSqY7fbdEb5c4s37Kv
5RPSUAgD/fFKLirpIks7vnRX8uRxOqfdFcl/5spZ2nyw1rWVeqlp0a52maZf20/dTL9QfLz4x+PbUw6DOmfqi402
SpPTvnpc5WkveAABEvfxDRfrw+92SpJ+d0rHmHY2RFpd/GXHaL45P7fFjiU2k/ld6f19R/YTnvsnOq1uwA4AXRSl
Ev3UfeH92HV7rbp+VzxVV3dA9H1/BakltRSl6SgGwOneZtOMrSdJre9tJ8q6KZzU/uex6fW+6KNkUa8Pbf410cwA
ADfDK53lye4xO65Kpnm3Sa95x48dS/jeSM1U67ZbaT+obwrf2PenwripPZzZJksTwPaA2FKUQ3faulwo2eJcM7za
ixt22HzimzfuOymG36bSuNc875StKrdp2UIeLXdXvlNLc+7P4UNDNBoCYtme15DoqV0ITrXO3V/fWaerUokmkWxV
2bZulanPviZKkduteUPHh/RFuEQAgGKVlHr38xTZJ0rVDOte+s6+X1KAbpCYtat83q4+Ufbpk3NKqF6o8zfA9oG4
UpRDd1r7r/dl1uJRc82S2vlX3BnRoqoxaVmHKzkxVpxapcnuMPt9Uw4cLekoBsLry+aR+SOwlj+w634K9pHzOG3u
DfrB1VJqO6etXH4x0cwAAQXj/f/nad6REWRlJyulTS07b8pmUt0xyJEpn3F6/k/snPH9e8rgrPeUfvkdRCqgRRSl

Et7XveX/2Gl3rbkvKh+6dWdfSrqowhG9jDUP4mFMKgNVt8668t+BYZ0kNnE8qxqUkOXVw8K8kSb23vqjv12+IcIs
AAIGas2yrJGncqZ3kdNTyEdjXS+qkq6WMtvU7+YljvCMtDm3zDv2rwLf63v4jFKWAmlCUQvQqzJd2rPD+3nNUjbt
5PEZL65jkvKI6i1L0lAJgdeU9pZaXnqCWaYk6KbtZZNsTYSdfeIO2JnZXU9tRbf/3ndp7mAlrASBWfL/zkFZuPaA
Eu01XnVrLqq07Vko/fCLZHNKZd9T/As4UacA47+8rnq30VCbD94A6UZRC9Fo3z/uzwylSepsad1u767AKjpYqNdF
Rrw9OZ5zQUjabtHHPEeUfKqq6g6+nVOlh72S/AGAlR/dJ+zdJkr7ydNO5vVrLYbdFuFGRZXMkqOW4WXLLrgs8i/X
EM0+ptMwT6WYBAOrhhfJeUiP7tVXrjOSad1w83fuz3+VS886BXcQ3hG/Dh9Kh7f7NDN8D6kZRCtHLV5SqY+iebz6
p07pkKjGh7rd001Sn+rdvWn5sQdUdKi77ymTnAKxmu3fo3hZbexUqzdJD9ypq0vkUHenv/dBx3f7H9MBbKyPcIgB
AXQ4dc+nNr3dIkq4d0qnmHfesKZ/L1iadNSnwC7XsLnU+SzIe6as5/s0tylffO3CsVB6PCfy8gAVQlEJ0Ki6UNi3
0/t7rolp39Q3DG1qPoXs+Z3YvH8K3YW/VJx1OKTGtvB0H631OAIgL27xD975wdVNSgt1/v4TUdPQUFae0UUf7XmW
tekwvf54X6SYBAGrx6sptKnZ51KtNugZ3al79Tq5i6b3feH/vfbHUqmdwF6s44bnLOxqjeRPvAkxuj1FhTSt/AxZ
HUQrRaWOu5HFJLXt4v3moQUmZW19s9q6iF8gHpzO7eSdEX7KxQMZU860F80oBsKrynlJfme46s1tLpSYmRLhBUSQ
pXclj/iZJmuCYp5fenqcVW2pYyRUALGbHwSJNfGmlZi34QQePlUoej3eepo//JL0yTsr/Jqzt8XiMXljuHbp33Rm
dZbNVMxTd45ZenyBt/UxKTJfO/X3wF+x1sZTRQTqyS1o+S5KUlOBQerI3jzKED6geRSlEJ9+qe7VMcC5Jq/IOqsj
lVsu0JPXMSq/36U/u1EwpTof2HSnRut2Hq+7gX4HvQL3PCQAxz13m/QAh6StPd51/IkP3qug1Sqb3xXLa3PqT41+
65YUV1c9PCAAW8+iH65T73XYtmz9X8x4cp0MPdJf+da53Rbt170nPXyztXBW29izcsFdbC44pPTlBY05qV3UHY6T
3Jklr3pEcidJVLwffS0qSEhKl8/7g/X3J37xzNEpqwWTnQK0oSiH6lJVI6+d7f69r6N4G36p7Lar/9qMGSQkOndo
ls9I5KqGnFAAr2vO95DqmQpOiDaa9zuvVOtItikq2kX+VSUzTyfaNuqB4nibNDe+3/wAQbQqOlGjzt5/p86SJmpP
4kMbZc9W0bJ+OmGR9nnyWDmX2987VOmeMtOOrsLRpztItkqQrBmdX3+t3wTRp5WxJNumnT0tdhjX8ov2ukNr0l0o
KpYV/lVRhsvMjFKWA6lCUQvTZsti78l1altR+UK27BjOflM+Z5cf4zlGJv6cUE50DsJCNH0uSVnp6qH92Zu2rFFl
ZRjvZzrtPkvS7hH/rh00b9fmmahbOAACLmLtim+60v6JM2xGZtCzt6TFO/2w3Tae4ntSVB2/RGTt/re/svaTiQzJ
zLmn0wtQXm/drwXrv3LHXnF7NBOdf/Eta+JD399GPSieOCc2F7XYp50/e31c8IxX8oMzyyc4LjpaE5hpAnKEoheh
Tceievea36KEil77dflBSkEWp8jmoPt+0XyVl7spP+npKMdE5ACspv//megZrRG96SdXqlJuk9oOUbivSfc7n9c9
PN0a6RQAQEW6P0dKlSzTM8Z2M7LLdNF+tx83Sbb+YqI/uukC3nH2CElIy9LNj/6cVnh6ylRxS8bMXa8f3S0PelqJ
St/707mpd+dQyGSOd37u1urRsUnmn79+Q5v2f9/ezJ3vv56HU9Wyp2wjJUyZ9dP/x4Xv0lAKqRVHK6lzF0opnpe0
rvOOqI83jkdbO8/5ex9C95ZsK5DFS11ZN1K5ZSsCX6pmVrpZpiSpyubUq72DlJ309pRi+B8AqCvOlHSskSbnuk5l
Pqi52h3TxYzI2h0Y7vpD54VOtymMeQgDW88naPRp17C1JkqfnKKl5Z/9z7Zul6HcX9tLyyefp3rGnamrTqfrS00P
J7iNK/89P9ceZc/TAvDV66fOt+mzjPm0/cExuT3CfSZb+sE8XzFikZ5ZsljHSpSe316OXn+R90hhp+0pp/u+l138
hyUiDb5SG/65hwddkxFTJZpfWvK1+Zq0kJjoHasKSOlY3//fSl//y/t6ql3TS1dKAn0lpEfqGfOdX3hUrEtOlLmf
VuJvHY/Tqiu2Sjg/DC5TdbtPQbi311tc7tWTDPp3etcXxJ+kpBcBq1nm/EPjK001JzdsFtHiEZbXpJ9upP5c+f0J
TEmbrr58M05PXnxHpVgFAWL2+5Fv9zbFYkuQ449Zq90lJdGjcaR111anZWraml9a/dY16lPxPv939O/15x9V62j1
cnvL+Ek6HTWkOh/62fomSEhxyJtiU6LDL6bCrWapTHTNT1bFFE3XMTFWnzFQ1TXHqkfnr9NLneZKktk2T9cDYfjq
nZyvvMMElb0jfvyUdyjveoBPHSKMekQKYkzYgWSdKA6+RvpqjEdv/od/rd0x0DtSAopSV7Vnj7SUlSQnJ0t61Uu4
fpI/ul3pc4L2R9hhZ6xC6kFv7rvdn9xFSQlK1uxhjdP873+ujNbvlsNs0dmD7oC/nK0ot3rhPv72gwmob9JQCYDX
lQ/fmuwfrJwPaBbR4hKWdPVll3/5XJxTlq/OGOfp+Zx/1adc00q0CgLDYtPeIOm/9r5KdLpW26qvEjkNq3d9ms+m
ME7tIJ3yg4ud/qoydn+uvzn/p1iafarrjRs0r7CKX2+iA26YDBccCbs+Ng5rp/7rvUsqGv0jvfywd2nb8SWcT72e
cPmOlXqO9PV4b0zn3St/9V1mHvtWF9i+1/2irxr0eEKMoSlnZh/dKxq38tufrQM4M9Sr4WPavX5S2f+n9xnzdPKn
tSdLIv0odT6v5PId3y77scQ3evFT2z9ZJ2adI7QYeL+wEwjefVK/RNe4yPXe95izbKptNmn7FAA3s2Dzw65Qb1r2
VbDbpm20HtWTDPv88U/SUAmApxYdkNi+STdJ8z2A9Ozg70i2KHSnNlJAzVXpron6V8Lr+kjtWD1x3QaRbBQBh8fK
yTbopwbtqduLQ2+rf8ygpXck3vit98ZS08CF1Ktmox3SP/jZgrLYP+j+9sWyjTj19iIzsKnF75CrzqNTtUcGRUm0
tOKa8/ceUt/+o8vYf1Yll6zSmyfe6tNkGpa/+Vvrec/w6FQtR3c6XElMb4W+hBultpDNulxY+pN8lvKIb9p0ht8f
IYedLH6AiilJWtSFX+uFjldkSdOWW0cp78lu1Su+g4T0e1cW9CnV64ftK+uYFKf9r6dkcqd/l0vlTpKYVeiUV5ku
fPSatfE6OsmK1l6QFXx5/vmUP7+p5Xc+WTrxEctayipMx0pq3pX3rJbvT21OqGv9atEn/+MQ7mezUMX015qTge0l
JUpumybr29E56ftlW3f36t5p/5zDvkrG+glphvrdt9BgAEM825MrmcWmjp51adu6rzj+eFBa1G3CVji1/Rk12r9T
pG/+mjXvOUKfmrFwIIL4dKy3Toa/+q7a2/SpJbqmkvpcGdoKEROmM26T+V0qf/lla+bzsq99Q9vr3NSrzHHWxNVd
Cp9O9+/1YYb70zSsyq16Ubf8PUpkk34LaLXtKJ5wrnXCO1Pms8BaifuyMX8mz4jl1Obpbww+/p3e/bfjnFyDeUJS
yIrdL+vAeSdLTrguVZ7KUmujQ3sMl+u/K7fqvJIf9TJ3b/kz9yv5v9d39tmzfvertxXTmJKnvpdLyWdJXcyS3d2l
TT/vBWuvpol7Ny2Tf+ZV0cKu3wLRvvfTNK95eWYOu965ukdGuQlvKpNVvSktmSLu/827rcYGUXHXow7+/yNNf5q2
RJP3fBT01vrrlXYPwfxf20kdr9mj7gSI98uF6/fHiE709xBxJUsEGafNCb2ENAOKUWfuev5fUlfSSCpzdrtRL/ib
Pk8P1E8cyzXz3Vf18/PhItwoAGtVbX+/UOM97kl1ynvbzGqfeqFNaK+nix6TBN0kfTJZt6xJ13/O+9ML7kjNV6ni
61GWY1HmYVLhDWvWitDFXMh7ZJG9vqJ4XegtRXc+p/CV6pCWlyX7OZOndO3VHwmu6PneERvdrqwQH640BPhSlrGj
Fs9K+9dqvDD1edomuG9JJ94zurRVbDmjBuj36dN1ebdxzRLnbpFxdqT62UzXFOUeDXeu832J8+ufj5+o4RBr+O7m
zh2rD+++r+6hRsjud0tF90o6V0rbPpW/+7U0gix+RPpsh9f6JdMoE7xxWS/8uHdjiPZeziTT4BmnYb6s0+b1v8zX
5DW/R6pfDu2ri2SeE7K8jLSlBfxnbV9c/96WeW7pZFw1oq5M7tvK25fMnpAUPSl2G01sKQHwqK5F7/XwlSFriOE3
P9Gsb6RbFprYDtL/31Wq55kWdv+URbd37k0i3CAAajTFGXyz+UFfZN8ptc8pxyk0NP2nb/tL176rs+7e166PH1b7

0B9mO7ZN++MT758c6DvHOgXviJVJSWsOv31gGXiv35/9S872rdemh5/Xm1wN12aAOkW4VEDUoSlnNsf3Spw9Ikh5
xXa6WLVvp7pG9lZTg0NBuLTW0W0vdO1ratv+YFm/Ypy82F+jzzcm67NAf9RP7Mt3tfFntbPu1zH2iXki6UqX2oeq
3uZl6F+3T3iJpd2Gx0lKMkpzNldQtR/YeF0hn3+OdwPyLp6Stn0nfv+7945OSKZ1+i7dQlZrp3+xye/T5pv364Pt
8zf1ym4yRrjq1o+6+sFfIJ+A9u2drXXpye73+1Q797r/f6t1fnamkoXdIK56T8pbRWwpA/Nq8WAmuI9ptmqnLgLO
UktjIE7/GsZYX/0mH172tHtqhd96eLrWtZT5GAIhhX+Ud0DkHX5McUlmfn8qRFqJJvG02mZ6jtPIHKWvkSDkPbJQ
2L/L+2brE23Oq/5XeYlTL7qG5ZmNzJMgx6iHp+Yt1jeMj3ZSbqzEnXScnvaUASRSlrGfhQ1LxQa3xZOtVz9n69+U
Dqv0Akp2ZqnGnddS40zrKGKPtB4r0+eaT9M8fxmjH1g1atL+ZjEvS2j36aO2e8qMS9OevF1U6T6LDrvbNU3Rmt24
adtpzGnreTqWuelr67r9Sk1beyf8GXiMleucvKXa5tWj9Xn3w/S59vGaPDhW5/Oe6eEA7/fmSvo22ItQfRp+oRev
3asOeI3r80x80aUQPeksBiHsl37+tJEm57kG6/JTQDIu2rNRMFZw+WelLJ+uc/Gf1XnrPuo8BgBj01qIv9Uf755K
kpKG3Ns5FbDYp60Tvn9Nvju15XrsMU1nPi5Ww7h394uhT+u+Kc3XVaeRcQKIoZS1718l88S/ZJP2pbLwmDO+hQZ3
qXrnOZrMpOzNV2Zmp5V1NT9WRkjKt3lmo73Yc0v92HNK32w9qW8ERlRmbPOb4saVujzbvO6rN+47qheVblWC36eS
O1+is03+ttBSndhUUa+dr67XrUJHyDxVrd2GxXO7jJ2jRJFE5fbJ0QZ82Gta9leyNuFpF8yaJmvKTvrr15a80a8F
GjerXRr3oLQUgnnk8cq/xrnq6OmOYru5QdT4/BKbz+Tdr45fPqZtrvbpufl4yP4t0kwAgpPYUFqvd+heV4PDoSNs
hSmvbPzwXjtWCVLmEC/+isg3zdYZW693cF3TpoMmirxRAUcpSzPzfy2bcynUP0v7WQ3TH+cF3eU1LStCpXTJ1ahf
vcDuXy6V58+Zp1KhRstkdKinzqNjlVpHLrdU7C7V4wz4t3rBXWwqO6Yst+/XFlv01nrt9sxTl9MnShX3aaHDnzLA
umzqqXxvlnJil+at363f//VavTxwqx6DrpS+ePN5bCgDixY6VSi3Zp0KTou6nj2q0nqiWYrer+IJHVfLOWJ1StlL
fvDpVA67+S6RbBQAhYYzR/a+t0F/s3jme0obfHuEWxZDmnaQzfiUteVgTXc/pv8sv0xWnhW6eXCBWUZSyim/+Ldu
G+So1Dj3kuVqPXTFASQmNM29IgsOuBIddTZK8b68OzVOV06eNJCmv4JgWbdirZT8UyMioTUaK2jVLVpumyWrbNFl
tm6aobdPkiH0wstls+tMlfbVsU4G+2X5ITyz8QbeeeYe0cnZ5b6lFUvYZEWkbAITavhWvq6WkReYkjRnUOdLNiRt
9Bw9T7pq7NOKHB9Rv/ePasnSQOp8R4FLpABCFXvk8Tzmb/qLmjiMqTe+oxB4XRrpJMSVh2CQdXfGCOhTv0pFPpqv
45L9HuklAxNFj0Aq2fCbz1m2SpFnun+iS84apT7vIDNHo2CJV15zeSY9ffbJmXj1If7z4RE04q6su6t9Ogzplql2
zlIh/U5+Vkazfj+4tSXr4w3V6YMkheQZd531ywYPe8ewAEA/WvitJ2t3ufGU2SYxwY+LLsCvu1DzHebLbjFrMv1W
Htq+JdJMAoEE27T2ig/Om6BLHUnlsCUq85O+SncUxApKYqqSR3t6z17rf0NuLv4xwg4DIoygV7wp+kPuVcbJ5XJr
nPlUL2tykm4fTTbQuVwzO1q/O7SZJemrRJv1mxzkyjkQpb6lsWxdHuHUA0HClu9aoZUmeSo1DPc4cG+nmxB273ab
Dva7Wt/ZeStcxHZ59pdxFhZFuFgAExeX26O3nH9ZE+2veDRf9TTrhnMg2KkYl9P+p9mQOUoqtVC2XP6ASd6RbBEQ
WRal4dmy/Sp6/VI6Sg/rac4KeanGXnr7+VCWw/GidbDabJuX01D+uGqikBLve+MHoLfsISZJ90V/pLQUg5v2w6D+
SpJX2/jrjxK4Rbk18Sk5MkPPKOdptmqtD2Vb98K9ryR8AYtLrr72sWw//Q5J05NRfyz7o2gi3KIbZbGr+07/JLbs
uMEv1w8b/qdhFZQrWFVR1YubMmerSpYuSk5M1aNAgLV5ce8+RhQsXatCgQUpOTlbXrl31xBNPBNVYBKCsRIefv1J
JhVu03bTU31pN0fO/OFst0pIi3bKYcvGAdnr15iHKykjStMMjVaoE2bctV9tDKyPdNCDqkSuim3PjPEnSkS4XhHV
BCavp1rWr1g6bqRKToB77P9XG16ZEuklAVCFXRL9vv1qukd//n5w2t3Z2GKW0C++PdJNinrP9AG3rcrkk6XdHH9Y
nj47X8u/WRrhVQGQEXJSaO3eu7rjjDt17771atWqVzjrrLI0cOVJ5eXnV7r9582aNGjVKZ511llatWqV77rlHv/r
Vr/Taa681uPGogTHa9/IvlL77CxWaFE1v+Sf98+cXqmmqM9Iti0n9OzTT27edqTbZXfVy2bmSpFM3/11Fjw+Ta/m
/pOJDEW4hEH3IFdFrzarF+uLhMepW6v3Pb++zfxbhFsW/4eeN0oedfitJ6vrdDG2ac6uObV1JrylYHrki+hXu26G
W74xXhu2YNqf2U7vrnpPsjLoIhc4/e1Q7O4yU3WY0xj1fJ/73bL0xc7L2HToc6aYBYRXwHWX69Om66aabNGHCBPX
u3VszZsxQdna2Zs2aVe3+TzzxhDp27KgZM2aod+/emjBhgm688ssTtUkn+am15ZZJabnpTZcaux1v
9QX/+5RVKT6Yg1RBZGcma+4vTtbb3r/S6+0yVmARlHFwt5we/VelD3bX1mWt1eN1CqfRopJsKRAVyRXQxHo++++w
9fT3tPPV+6yKdenSBJOnLNlepQ8cukW2cRVx47e/0QcpFstuMum56UanPnavtDwzUt//5kwr3bIt084CIIFdEKVe
xjqxfpLw3p+rIkxeqndmj7ba2avXz1yRncqRbFz+S0tXquhf08Qn3amdKD2XYijR2z0wd+dspWvTO89p34KAMX17
AAhIC2bm0tFQrV67U3XffXffXWl7Tk6Oli5dWu0xy5YtU05OTqVtF1xwgZ555hm5XC45XC45nfUvlnzz0ctKa5ISSJMjzu3
2qGTbJn2Tu1+OBszlZIyR8bhljEdye38aj1s6vEuJ+9ep+ZGNalu2TUkqU+fyY15q+Svd+cublexkVYxQSHY6NG3
cmVq47hn96r1P1OvYFxrtmq8e9h3qtO0t6ZW3JEkHbU1V4GyrI8ltVZyWLZPeVjZHomyOBMmeILvDITkSZLPZ5as
L+xccLP8l0isQSqF778YCq8R66HB4iqbkisAF+h6s6T+pxuOWKXPJ43bJlJXKuF0yZSVqvuV99Svzrv7mNjZ92+w
8tbzwLp3S+7SQxoGaJSbYNeS25/Tq2y+qxcbXNLTsc3VwbVaH1Y/I/f2j+l9SPx1Lai1PYro8SRlSUoYcKRmyJza
RzW6XbHZv3rDbZJNdstnkyxSmQs4Id/6wyv1Tsk6s5IroFbLPFR6PjHHLeDySx+39TOFxy7Z/k5ruW6FOxWuVpjK
lle+/36Tp0E9fVofmWaEJBJUcyeipVlcs1rYlzyttyQPqbPLVeeWvpJW/0j7TVPsSsnQ4qY1K0jrIlt5WNmeSbPY
E2RzeP3aHU3a7vfxzRIXPFnyuiAirxBrKXBFQUWrfvn1yu93Kyqp8Q8rKytKuXbuqPWbXrl3V7l9WVqZ9+/apbdu
2VY4pKSlRSUmJ//GhQ97hUV2W/J8ykiL/jypQPSRpR+Nfp1hSgUnUJmVrS5sLlXPFbTpaeFDh+K+Fy+XSsWPHSV
QENB/CGJRr2bS2e1sOvvs32nD3l9p4VeL1OKH1zTY9aWa2Y7JroNqVXxQrQ6vkfZGurUNE673bjSwQqyFJd5CRmN
/60auCE5jvwf3mgR9l3mB2o34lTp27CFJKigoaLwLVsNKuaKmWM/NGSMz4if6Zlu+dnzxX7XLe0+9PevVsfQbKUZ
HbFjh/uljhVjJFdEtHO/BYknbTYa+t/fSnmYD1OaUS3R6u05hyxmWzBUHDir1pMtU2iNHi996UF13vKl0FSlRB9W
u9KB0bJ10INKtbRgr3D99rBBrKHNFQEUpnx9XW40xtVZgq9u/uu0+06ZN05QpVScCzf7bkUCbalEFkr6W7nww0g0
BEIUKCgrUtGnTRr8OuSIa/Vv63b8j3QgAMYBcYXWH5f1U/bGk6RFuC4BoFYpcEVBRqmXLlnI4HFW+vdizZ0+Vby1
82rRpU+3+CQkJatGiRbGRbXHTJ48WZMmTfI/9ng82r9/v1q0aBEQ8DUVhYqOzsbG3btk0ZGRmbk6jsUqcknVitUq
cknViPXTokDp27KjMzMxGvQ65InBWeQ9aJU7JOrFaJU7JOrGSK6KXVd6DVolTsk6sVolTsk6socwVARWlEhMTNWj

QIOXm5mrs2LH+7bm5uRozZky1xwwZMkTvvPNOpW3z58/X4MGDa+yOmZSUpKSkpErbmjVrFkhTo05GRkZcvyl9rBK
nZJ1YrRKnZJ1Y7Y28ag65InhWeQ9aJU7JOrFaJU7JOrGSK6KXVd6DVolTsk6sVolTsk6socgVAZ9h0qRJevrpp/X
ss89qzZo1uvPOO5WXl6ebb75ZkvfbiGuvvda//80336ytW7dq0qRJWrNmjZ599lk988wz+u1vf9vgxgMAohO5AgB
QF3IFACDgOaWuvPJKFRQUaOrUqcrPz1ffvn01b948derUSZKUn5+vvLw8/3u7dOmiefPm6YknntA///lPtW/qXvXLw8//5dunTRvHnzdOedd+rxxx9Xu3bt9Pe
//10//elPQxcFACCqkCsAAAHUhVwAAgprofOLEiZo4cWK1z82ePbvSsQp78ePbvKtuHDh+urr74K5lIxLykpfSffdd1+Vbs
xStaJ1SpxStaJNdxxkivqj/dg/LFKrFaJU7JOrOSK6MV7MP5YJVarxClZJ9ZQxmkzjb3eKwAAAAAAAPAjjTuDIQA
AAAAAAFANilIAAAAAAAAAIO4pSAAAAAAAACDuKUo3g/vvvvl81mq/SnTZs2kW5WSCxatEgXXX3yx2rVrJ5vNpjfffLP
S88YY3X///2XWrp1SUlJ09tln6/vvvv49MYxugrjivv/76Kq/x6efHpnGNsC09oadN0ymnKD09Xa1bt9Yll1yidev
WVdonXl7T+/sQaD6/rrFmz4Rh0nn5/n5+f4/3333++Xl7PeECuiP33IbniuHh5TckVXvHyesYDckXsq6/F++JBaD
00YgRI3T48OEwt7Rh6opTki688EJJ3/xyJJvB3T/X//b8J/0tKrsgTk4FrPuqNA/ImFwG1rDx5GCzkffSlGq2UEHAh
FTQDWAIB7OjPa3SuvDbOW5Bpd0KxQt6htvgEHbhVmgFZMhSHxFMQYdOy54gAm47yWY46BtfS0mUbnXzBvHhcm7s2zbSfW0a7yeCbFDaX4QpcoJDvvdwfJSsYzoW/0Y0pmGd4eY5DzdYMKmnf/kKsOqt+3y8Flvi2bcaRDbSKr+xgSK0vPfUIqLt5HAd35Zg5WH6nGYU9bv8o6WABjFVPLYv9PBCg8N9NDAFnn1
s5YMpkIhFIDjN3xnWppHuOMgZ3Gx9j60ttqPoe02NTuk7hkLzhH1LD0E/JaaaXYLDEfANDAKPbfmjc2MVl9tXuNYjQc+8gMJuu6SWlWrCVK4ZB/5fFS+Cbo
Dv3WhxgjMHZgcY7vPq1enF09PLJBn/mK6r6mOXmZbjfMvlfn1lUKDJz62sLxiXdUXZ69pz1hnX35d7f/AMdqrE7UfYNAM7hgTr/4XcqMmqjzL6ZT0DqJTO7rGsk
Z2Ytc+Fq5lzuZHbqCXH0X90CmX4EEhjiASvPy/uKEJWKgmp8vHi8C7BzMUGeZfTuqS+ZD8Gdz/ZKJIUIeYRF7jIcVRsMaSe7skq/pbzRWEN+MOBW8d3L+9
m4SXqb6ur79Z2cTt1HH6o2qG4X03F7mfiv2fyLLs/Z6aW+1exn3eK9Z+NdQivgg6rtzkvHXqDY5q+L/7d2tzurrC3O3TnchpUX2LZtV6Wzns8a9xzydRsnnz
rc5BJ+N5xFbVavXpVmJHm9u+b5FE9yaLX/nebo

FWYHAw6QUAsrkyZO1detWjRkzRoMHux++iYmJCg0N1f79+zV69GhJ0m+//aZvv/3W5cxCTEyMWlparOcnTpxwmcW
fPHmyPv74Y40YMeKa7rOWpEmTJqmnp0dVVVVuZxoumzFjhiIiIlRaWqpdu3apurr6qn0mJyertLRUxhjZbDavx7R
161a9+OKL2rVrV6/3l18+Y5OcnOx13wAwUJAVZAUAeEJWkBUYGPihcwSUxYsXq7W1VfPnz7f+zaGiokJFRUXq7u6
W3W5XcXGxli9fri+++EJHjhxRYWGh26WkU6dO1dtvv62DBw+qtrZWjz/+uMvZiQULFmj48OGaPXu1GNPrDYhISEqLCzUypUrlZiY6HJ5cG+ysrJ04cIFHT1
61ItP7ZIjR44oPz9fK1as0IQJE3T27FmdPXtWra2tVpv9+/crLCzM4zgAYCAjK8gKAPCErCArMDAwKYWAMmrUKNX
U1Ki7u1sPPvigJk6cqJKSEkVFRVkB8dprryk9PV2zZs1Sdna27na27r33Xk2ZMsWln7Vr1youLk7p6el65JFH9
c3jps2DBVV1dr9OjRys3N1R133KGioiJ1dHR4dYajtLRUc+bM0ZNPPqnx48frscce04ULF1taFBcXq6urS0VFRR7
7i46OVm5ubq+X/3pSW1ur9vZ2rV69Wg6Hw1pyc3OtNps3b9aCBQuBQuueqkvAAx0ZAVZAQCekBVkBQYGm7nyBlggCGV
mZiopKUlvvPGGv4fipqamRpmZmTpz5oxGjhzpsX1DQ4Oys7N7/cHE/+KXX37R+PHjvVVtbq4SEhH7rFwACBVnhGVk
B4P+OrPCMrIA3uFIK8JPOzk41NjZq1apVmjdv3r8KDunSPeWvvvqqmpub+3U8TU1NeueddwgNeueddwgCdkBQI
ZP3QO+MnmzZtVXFyspKQkffDBB169tqCgoN/Hk5qaqtTU1H7vFwBw7cgKAIAnZAUCGbfvAQAAAAAAwOe4fQ8AAAAA
AAAA+x6QUAAAAAAAAfI5JKQAAAAAAAPgck1IAAAAAAAADwOSalAAAAAAAA4HNMSgEAAAAAAMDnmJQCAAAAACAzzE
pBQAAAAAAAJ9jUgoAAAAAAAA+9xfzYX+Hq6orpQAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 1200x500 with 3 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"psd_r, freqs = psd(trials_csp[1])\n",
"psd_f, freqs = psd(trials_csp[2])\n",
"trials_PSD = {1: psd_r, 2: psd_f}\n",
"\n",
"plot_psd(trials_PSD, freqs, [0,28,-1], chan_lab=['first component', 'middle component',
'last component'], maxy=0.75)"
]
},
{
"cell_type": "code",
"execution_count": 34,
"metadata": {},
"outputs": [],
"source": [
"def plot_scatter(left, right):\n",
"   plt.figure()\n",
"   plt.scatter(left[0,:], left[-1,:], color='b')\n",
"   plt.scatter(right[0,:], right[-1,:], color='r')\n",
"   plt.xlabel('Last component')\n",
"   plt.ylabel('First component')\n",
"   plt.legend(cl_lab)"
]
},
{
"cell_type": "code",
"execution_count": 35,
"metadata": {},
"outputs": [
{
"data": {
"image/png":

"iVBORw0KGgoAAAANSUhEUgAAAkMAAAGwCAYAAACq12GxAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABhA0lEQVR
4nO3de3gUVZo/8G8RkpAICZAQEpJIwp1RUIRVwEEIyk1HGSOo4DIwK4zosIjoiIJf9GXB0UBcQ1xlHZZ0wroAiibq
O7ihi4g1UQEBGlGuQGBIDGUhwHBPonN8fZTXppC9V1XXXt+n6epx9Ipbv6VHV36u1z3vMeSQghQERERORR7exuABE
REZGdGAwRERGRpzEYIiIiIk9jMERERERexmCIiIiIk9jMERERESexmCIiIiIPI3BEBEBEREXkagyEiIiLytPZ2N8DpmpubcfToUmpubcfToUXTq1AmSJN
dHCIiIlJBCIFTp06hR0HxcoYjF2zd3xoP9feyZRxIvbWUbJ3GwSjASbrXLYWWq7xlEr1Ue8eVYuVbo2JcPpqwSPRuBu
EpEZDQwNyc3P91/FwGAxFoAyNpaSkMBGiiIyJyGTUpLkygJiIiIk9jMERERERexmCIiIiIk9jMERERERexmCIiIiIPI05Q0RERDbx+Xw4ffr
03c1wpfj4eMTcRFxRmyLwZDRETEREhFhNCoKamBt+fvRmN6L5FHkK9I5wGkE2PI0tcEpLMvMD+wd8j0YUzQlwtkb5dvkflo8rEwcFsOeKdMHS3Y4f03GoanMB1JbQGL0f9Y2X1eQSe/ITxhCyPc
03c1wpfj4eMTcRPmyLwZDREREFhNCoKamBidPnrS7Ka7WuXNnZGZmNnQdGZmRl0HkMEQERGRxZRAKCMjA0lJSWzGGBMQpuMLVhXRbDGOTzqq5EQAt9
//z1qa2sBAFlZWVHtj8EQERGRhXw+nz9QSktLs7s5rpWYmAgAqa2tRUZGRlSXzDhSiojItBZLAqfCVDiWGL7GDsHiYLFo0DZdwiwedGQVAtXFdpQWjpwAnNwnSBjLDKvVUa0VG9SgEUP0lzuCabWOr5qdfq5KDEXFVtE+l6eH9HQnm41qPZy7Lb4VJhp
R5l0xGCIiIrIBh8aiZ9Q5ZDBERERE4YWGeFQXTN0EfaVXESKUhpbxY1GCBAYDg==
C+9pFRG1xmIyIyGClpcCUKYGBEABUVcnbS0vtaRdRtMaMGYMFCxaovv+rr76KPnJ6CDtuTtPjrMZgiIjIQD6f3CM
kRNvfKdsWLJDvRxQtnw8oLwfWrZP/ddr76pZbbsGUKVNQWVmJ3/72t5g1axZ+/vOf292sNhgMEREZ6IMP2vYlSQ
EUFkp348oGqWlQF4eUFAATJ8u/5uX55yex++++w61tbWYMGECevTgU6evTogU6dOtndpJAYDBERGai62tj7EQXjhKHYpqY
m3H333cjOzsY555yyDSy65BOXl5QCA8vJyf/AzduxYSJKEMWPG4M//+nWlgM+rPe3wPBPuUGuS0SQu97xyPTDvVdQ==
ZycMiDcVKkjwUO3myubMXf/nLX+Lw4cN48cUX0aNHD7zyyiuYOHEidu/ejZEjR2Lv3r3o378/SkpKMHLkSCQnJ2P
OnDloaGhAcXExAKBr167mNVAD9gwRERlo1Ch51lio8ieSBOTmyvcj0sMJ0Q7EHDx7EunXr8PLLL2PUqFHo3bs37rr
rLvz0pz9FcXExEhISkJGRARAUAOeDIzM5GSkoKpCQkJiYmzZmZMTmTZmZSEhIMK+RGrBniIjIQHFx8vT3KVq8s3c
lQFq5kvWGSD8nDMV+9tlVEEkgX79Ad8bGXtducQIgyEioMVFgIbNgSvM7RyJesMUXScMBTb3NyMuLg4bN2NvvvuMpbP
aYBO7djTViU3CYiIyASFhXLOBitQk9GUMOnRYOHMvNaEAAvoM/8PBkNERKa
JiwPGjLG7FRRDtntDAU269fP9x00034xs1SAtg9z8xSS9+geXl2PIkCE4fvtehb6fewt69e5GWlob
U1FTEx8enBRETkMspQbHZ24xfS9tJFr8BBKEUkhcXt6TaFuJ7XBPQiLfwdxm4Ke6v6c6UgCbAFWfX+KVzWvTwdfJ9fOnPP1XYsOWiqmVaxDpAw==
OQDZsGF599VVIkoSdO3cG7Gv79u0YNmwYkpOTMXLkSOzduxcAsHr1aixZsgS7du2CJEmQJAmrV6827ZgABkNERET
uU1oKTJkCfPNN4PaqKnm7iQHRn//8tnUGDRqEzz77DL/97W9wCLdffTUGDRqEzz77DL/97W+xaNGioPu5//
7sXz5cmzbtg3t27HHv/3bvwEAbrjBtx5550050477zzUF1djerqatx5556m/7z6zp6y/HG/
NSFETLtpe7HWP2Cge.z
WLAAmTzZlyKxPnz547LHHgv5uzZo1kCQJq1atQocOHfCTn/wEVVVVmDNnTpv7Pvzwwxg9ejQA4J577sFVV12FH37
4AUlJSejYsSPat29v2RAce4aIiIjc5IMP2vYlSkhTa0rEspHCrmxY2YexvHSKwqAQMzp/
/PysrCwBQW1trUCu1cV0w9NRTTyE/Px8dOnTA0KFD8UGYF1F1uZvtf69tVXX1nYYiIiIgNVVxt7P430OeecckL8TQkC
SpDbbgomPj/X3lMc3OzAS3UzlXB0EsvvYQF44ChMmjQR+ve4FCcCbg/vvvvv3+sd6cdq6c8vRI9cdvvUdvX4/mA5P4z9h6JAh41Hx1X3QxWUZ
aTEREZLAfe1EMu5+BBgwYM8//vr+8dd2bd08n4SEBPh8/sQgW9o5Bdfdqq+6SWs1/Zb2RtBS7B8FwtfbdPmDsGqbOxt7VeuuzS6fYMI=
5+OMf/xj2cRkZGcjMzPTf4iycdkhERGSoUaOAnBw5/sbsdcNAqySx9R7CZrurIN6Cb40e0/9Xy/ZFS2Ihp60qNtjuK1Stg4Zxu
bFaK9QeTl5aGiogI7d+7E8ePA4IrM7gmGGpqasL27dsxfvz4g3jx4/H5bwz52yJAhyMrKwuWXD
sbERDQ0PAjYiIyDHi4oAnnpD/3zrcEQCzNH4eudLSekOKlJQUvP7669yicuvPBC3H///XggQcAI9EBHM3bPF8ycPF9B==
OREFBAbp164Z169aZ1WQALppNdvz4cfh8PnTv3j1ge/fu3VFTUxP0MVlZWXj22WcxdOhQNDY24n/+539w+eWXo7y
8HJdddlnQxyxduhRLliwxvP1ERESGKSwENmyQZ5W1TKbOyZEDocJCU562vLy8zbbbWOUEjR47Erl27/D+vWbMG8fH
xOPfccwHItYpaP+bCCy8M2JaYmIgNGzYY2PLwXBMMKYIlZoXqeuvfvz/69Lk5LxYzmbOvWwCtrAGCBdNFAZp6Vr9fJr1Hb1vyJg+2bFwk6iLXNCqt/XSNj6U87D5Vpn2wtcqFR+C8Mq8rLgg1+SBCAc==
uXOj/uaGhAbm5uQa0nIiIyECFhfL0eRsrUAfz/PPPo1evXsjOzsauXbuwaNEiXH/99UhKSrK1XeG4JhhkT09HXFx
cm16g2traNr1yF4QwfPhwvvvPBCyN8nJiYiMTFRdzuJiIgsExcHjPT8vK56uP95uHFa8HgcD2uBJ1Ewp4lb5xz3CzWdOz+FrHs
3KyzX5AwlJCRg6NCh2LhxY8D2jRs3YuTIkar3s3s2PHDn89AyIiIjLW3XffjcOHD+OHH35ARUUFHn/8cSQnJ9vdrLB
c0zMEAAsXLFsSMGTMwbNgwbNgwAs8++yyOHDmqqqvD8888DpCHuKqqqqvD8888DAFauXImhK
UlJTYeRhERETkIK4Khm644QbU1dXhwQcfRH19PYc4//zz46//jD/+zzyyyezi5c8g9UYG6gNSFHIFFQu/v2K/YpzNrxS09EyOeSeRQ/RA9/jfZ9n6Y/p9W8
e3njjDVx55ZV2HQIRERGA0MUIST2jzqEk+GqE2jzqEk+GqE1dDQgNTUVNTX1yMlJcXu5hARlmB/MC
XV4fa2lr069evTQ1BLddvV/UMEen8hzlusgUReVhcXBw6R6kD1BM3PR...RKCjVWnHyZGn/A5wmzAAp8SJlYB/x8uYnpfXFmZKUmRbwWrVkpYbF2l/K+6q1k5b77UHJUR92AQ==
xrbQ0eBmOJ54wrQwHEVFEymrsdi1MGis6d+5syMr2HCaLgMNk7lVaCkyZIi/g3JLyBWzDBgZERGQvn8+H06dP290
MV4qPjw/bI6Tl+s1gKAIGQ+7k8wF5eYE9Qi1JktxDVFHBITMiolik5frtmjpDRFp88EHoAiQe4sqK+X7ERGRtzE
YophUXW3s/YiIKHYxGKKYpLbIObREOHtOZhIBJTBo1Ss4JqqnqpmB0ANnM0ZGjXK+rYZxayaSAsxFQERew54hiklxc
0eeDs7DGF8vPKle69yJeWygniBQXA9OnyjJYpjecc0Ns3d2F7dlx87R7eRaKsXvaI+ms2LBBw3Ub8v5u6bDi2vj
ARe9+fT6gvBxYt07+1xERHbh1PoIOLXe/WJp2MeskgF698uilkTkVJxaT9RCXBwwZGwbwZr8r1sDIcC8kgF
69mtWDxURkdUYDBG5iFklA7Tu1+eTe4rGSQcfgpsgjnhwoDGwZEbC6ilJAwTtcC1JAG
5udpLBmjdL4taElEsYTBE5CJmlQwBn1Ci5KNRMNMxzBMxqxaCoZ3GW/ZvVQERHZgVPrI+DUenuRsDzwIv+DUe
AcnMtJyJyPy3XbwZDETAYIVJ2hOWLwZCBGAwRhdeyJ2n/fmDDXqjCL7segiEVlGKWqZmAg

sXswijETkPgyGiChqaoswNjVxHTMicp72djeAiNxPbRHG7Gzg+PGz2zmEZp9YWrOPKFrsGSKiqKktrtgyEAI4hGa
X0lJ5Yd6CAmD6dPnfvDy+DuRdDIaIKGp6iytyHTPrcYFdorYYDBERfL7ocnkiFWEMh+uYWYcL7BIFx2CIyOOMGDI
Jt5yHWlzHzHxcYJcoOAZDRB5m5JBJqOU8unVT93iuY2Y+LrBLFByDISKPijRkIoT8ey1DJoWFwOHDQFkZsHat/O8
333AdM6fgArtEwTEYIvKoSEMmgPz7hx/Wtl+lCOO0afK/CQmhh9CUn1eu5LRuK3CBXaLgGAwReZTaoZCiouhnGIU
aQsvJ4YKuVgqX28XAlLyMwRCRR2kZCjFihlGwIbSKCgZCVmNgStQWF2qNwOkLtbKKLOnl88mzxiINlSnKyuRhL4o
N/NtBsU7L9ZvLcbhYaamc4MoVws0VqxcNZcjkuuvU3Z8zjGKLkttFRBwmcy1WkbVGrC9bUFgILFmi7r52zDCKthg
kEZEaHCaLwInDZJGGNyRJ7iGqqIiNHgw7+HzyLKqiora/UxJNYyW/wqnvJ/Z8ElE0tFy/2TPkQqwiay6lNyhYIAT
E3rIFynCZJDlnhhF7PonISgyGXIhVZM2zYYOcQxMpqTjWAk4nzTDi+llEZDUmULsQq8ia4+WX5UKBWsRSwFlYCEy
ebH+yuJaeT7cmAMdqUj6RWzEYciGlimxVVfBvz0qOB6vIqldaClx/vfbHuS3gjHQRdsIMo1jv+WQuFJHzMBhyISX
HY8oUOfBpGRCxiqx2yrCMFk4OOEMFPKEuwitWyIupOqWXIpZ7PpVcqNZfYpRcqFhJyidyG84mi8CJs8kUwS5uubl
yIMQ/qOqVl8tT5rWQJGdeuEIFPNOmAcuWBe9JbE1rL4XRQz7K7LZIPZ9umy3p1Fl7RLFKy/WbwVAETg6GAOYeGGH
dOrmGkFpOHdII1evQuvcwknClA1q/344fB+64w/ghH+VYgOA9n04MRCNRG3Sz0jeRMViB2kOckOPhJHqCQy3DLUu
WAPff77yAU80MLLWEkIOOBQvkhGrlWIP1OgVjxJCPMrstWC+XW3s+Yz0XisjNGAxRzNCbmBopIR2QA4IXXzzbW+E
0kWZgadV6xlaoXqdQjw0WTGnllNltRonlXCgit3NdnaGnnnoK+fn56NChA4YOHYoPIhR6ee+99zB06FB06NABvXr
1wtNPP21RS8lK0RTpUxLSgbZFBxXr1jk3EALM602org7f6xSKUXWYlJ7PadPkf90aCAFng+5Q7zFJknP+nJiUTxT
rXBUMvfTSS1iwYAHuv/9+7NixA6NGjcKkSZNw5MiRoPevqKjAlVdeiVGjRmHHjh247777MH/+fJSUlFjccjKTEUX
6QhUdzM0FSkqAqVMNa64pzOpNyMqKrteJQz5nhQu6OQuUyF6uSqC+5JJLcNFFF+GPf/yjf9vAgQPx85//HEuXLm1
z/0WLFuF///d/8eWXX/q3zZ07F7t27cKWLVtUPafTE6jJ2MRUMxLSrUhyjzQDS6uWM5vWr9eWYN6S2mRgL00E4Cx
QImvEZAJ1U1MTtm/fjnvuuSdg+/jx47F58+agj9myZQvGjx8fsG3ChAl47rnncPr0acTHx7d5TGNjIxobG/0/NzQ
0GNB6MpORialGJ6RbVWBPTe2p668HXnpJ/T6VXgo9vU5a6jB5rQhhrOVCEcUC1wyTHT9+HD6fD927dw/Y3r17d9T
U1AR9TE1NTdD7nzlzBsePHw/6mKVLlyI1NdV/y83NNeYAyDROTUy1erHRSOuLTZ6sbj9paYEzwSLlurSmZcjHqwu
yxlIuFFEscE0wpJBa/UUWQrTZFun+wbYr7r33XtTX1/tvlZWVUbaYzObExFS7FhstLAQOH5aHp9aulf+tqJC3qw0
G580L7JFRk2DektrFXbkgKxE5hWuGydLT0xEXF9emF6i2trZN748iMzMz6P3bt2+PtLS0oI9JTExEYmKiMY0mSzh
xeRI7FxsNNdSnpoQAADz4IDB4cGAwE6ruT24usHy5vuU8vLAgKxG5g2t6hhISEjB06FBs3LgxYPvGjRsxcuTIoI8
ZMWJEm/u//fbbGDZsWNB8IXKvSENEVueeOLHAnhI0qkmwDtYjE6rXaepUfUM+TjxHRORNrukZAoCFCxdixowZGDZ
sGEaMGIFnn30WR44cwdy5cwHIQ1xVVVV4/vnnAcgzx37/+99j4cKFmDNnDrZs2YLnnnsO69ats/MwyCROSkw1O49
J7+yrwkK5inZRUej7hOuRMTLB3Km5XkTkPa4Khm644QbU1dXhwQcfRHV1Nc4//3y8+eab6NmzJwCguro6oOZQfn4
+3nzzTdxxxx34wx/+gB49euC//uu/cN1119l1CGQypyxPEmlIKppV76OdfdW3r7rnMbtHxsxzRESkhavqDNmBdYZ
ILzMWGw23GKvafaqty/TOO8Dll2trn1axuCArETmDluu3a3KGiNzG6Dwmo2ZfqZ0qP3Om+VPbnZbrRUTexJ6hCNg
zRNEyqrqykZW2Q/XItBRN74zWY/ZSBWoiskZMVqAmciuj8piMnH0Vaqp8S3pXn9eT0+SUXC8i8iYOkxG5hNGzrwo
LgdWrw99H6+rzXq0oTUtuxmAohvl88tDKunXyv5FySbTen6xlRqXt2lp191PT28SK0kTkVgyGYlRpqbyKeUGBvOJ
4QYH8c6hv5lrvT9YLtyyG3krbRvY2aakoTUTkJAyGYpDWoQoObbiH0bOvjOxtYkVpInIrBkMxRutQBYc23CfcYqx
aGdnbxIrSRORWDIZijNahCg5tuJMy+0rremDBGNXbZEZOExGRFTi1PsZoHarg0AYBxqzrpvQyTZkiBz7BKkprzWk
iIrICg6EYo3WogkMbzmV1IUIjav2Eql+UkyMHQqwoTUROxArUEbitArXPJ88Ci7T4ZUWFfPHTen+yRrSLsdqNFaW
JyG5cm8zDtCbEmjFdm6ITC7P7jMxpIiIyG4OhGKQ1IZaLZToHZ/cREVmPw2QRuG2YrCUulum+YzJyMVYiIi/jQq0
EQHtCbKwtlunGvBvO7iMish6HySgmuTXvxgmz+7hGHRF5DYfJInDzMJlXKTPkQhWTdPIMObtn97mxN81WbhuHJfI
QziYjT3NzVW07Z/e5tTfNNlzdmChmMBiimOP2vBs7ZvdxFptGjByJYgoTqCnmOCHvJlrRLo+hdfRGS29aLCXZ6xI
pcpQkOXKcPJlDZkQuwWCIYo6yYGikvBunLxiqd3afnrwft/emWYqRI1HM4TAZxRwvV9XWO3oTC71plrEqcuS0PiL
LaA6Gxo4di5MnT7bZ3tDQgLFjxxrRJqKoxUpVbS3Xw2jyfpTetNbBo0KSgNxc5/emWcKKyJHJ2USW0jy1vl27dqi
pqUFGRkbA9traWmRnZ+P06dOGNtBunFrvbm6e+ax1uEtt9ep33pHPQetzovQqAYEBlRIguSmINJXZ9Q+UF6L1vvl
CEGliSgXqzz//3P//PXv2oKamxv+zz+fDX//6V2S3/hpOZDO3VtUOdT1UhruCXQ/Vjspcfz3w97+f/bllgLVhQ/A
AbOVKXn/9lHHYKVPkACVY5Kh3HJbJ2US2UN0z1K5dO0g/ftCDPSQpKQlPPvkk/u3f/s3YFtqMPUNkNb1FI9X2DAX
bH3A2wHJzb5qlgnXd5eZGFzlycToiw5jSM1RRUQEhBHr16oVPP/0U3bp18/8uISEBGRkZiONfTKKo6Z2sFGkWXbj
9te5w4HVWhWjrHwTDaX1EtlAdDPXs2RMA0NzcbFpjiEj/9TDc6E0knA2uk9GRI6f1kRbsxjWMrjpD+/btQ3l5OWp
ra9sERw888IAhDSPyqmiuh6HyftSK1Q4H11wzYqVIFpmPCwkaSvNsslWrVuHWW29Feno6MjMz/XlEACBJEj777DP
DG2kn5gyR1YyYrLRpE3DFFdqfOxZTUVx3zeC0PoqEMw5V0XL91hwM9ezZE7fddhsWLVoVoUVSPdgsEQ2SHa6+G6dXJ
5GrWinQ3uVK69ZpiRnE2xQe8MCw8yddX6EydOYOrUqbobR0SRRVs0UktKSaxW5Xb14rOFhcDhw3JX3dq18r8VFQy
ESNsMC1JNc87Q1KlT8fbbb2Pu3LlmtIfIPfkdJotmspKWmWWxWkfI9UuIcVofBcMZh6bQHAz16dMH/+///T98PH
HGDRoEOLj4wN+P3/+fMMaR97juvwOk+m9HqqZWaZMpY/VYJPXDIpJnHFoCs05Q/n5+aF3Jkk4dOhQ1I1yEuYMWce
1+R0O5uXUE9YvpJhk9nIwMcTUBGqvYTBkDeYEmserw468ZlDM4oxDVUxNoFY0NTVh7969OHPmjN5dEPkxJ9A8ylD
btGnyv1658CtDhcDZa4QiVpPGySOinWFBbWgOhr7//nvcfPPNNcjcjR44AkHOFHnnkEcMbSN7A/A4yg9e
uGT6fPDy4bp38ryNnypExOOPQUJqDoXvvvRe7du1CeXk5OnTo4nTo4N9+xRVX4KWXXjK0ceQdzAkks3jlmlFaKg8LFhT
INaYKCuSfS0vtbhmZxqvdvibQPJvs1VdfxUsvvYTw4cHw4cRaOPIOrkJAZor1Weovvwxcf33b7VV
VcmpJLPaCERlJc8/QsWPHkJGR0Wb7P/7xj4DgiEgLs/I7OGzgEnyhdNuwQe4YCMbxxSWJHEJzMPQv//IveOONN/w

/KwHQqlWrMGLECONaRp5jdH4Hhw3OcnSswRdKt9JSYOrU8K8nJx8QRaZ5mGzp0qWYOHEi9uzZgzNnzuCJJ57AF19
8gS1btuC9994zo43kIdFUXW4pVM0iLw4bOLqQpVtfKAfUK1CWG1GLkw+IQtNVZ2j37t1YtmwZtm/fjubmZlx00UV
YtGgRBg0aZEYbbRUrdYYc8LfbMl6oWaT29XR0IUu3vlAOiS7VFpVUsLgkeQ2LLhooFoIhh/ztDmBmcOb0ysPRHrv
a19PxsYbTX6hgHBRdrlsnjyqqkZvrvJiSyGymF11sbm7Gvn378OGHH+L9998PuJnlxIkTmDFjBlJTU5GamooZM2b
g5MmTYR8za9YsSJIUcBs+fLhpbXQi5W936wuiMgphR1qG2SkiTq5ZFO2xa3k9HV/IUuUL0FxV7Yx8J2VcKtj3Rxs
ylbWUmWBxSaIIhEZbtmwR+fn5ol27dkKSpIBbu3btt05OtYkTJ4rzzz9fbN68WWzevFmcf/754mc/+1nYx8ycOVN
MnDhRVFdX+291dXWanre+vl4AEPX19dE03xZnzgiRkyOE/Je67U2ShMjNle9nlZIS+XmDnO9Hq6ws3BtDG3vJW
VRf9cWkR77Fpfz7Vr1Z2HtWvNP/agVL5QU9LLAjbl5BjzPjGrvVa9sZT3Q7D3lHKLixNi/XpLmkPkOFqu35qDoQs
uuEBMnTpV7NmzR5w4cUKcPHky4GaGPXv2CADi448/9m/bsmWLACA++uqrkI+bOXOmmDx5sqbn+uGHH0fb3/Vll
Z6dpgyIy/3WfOyPdfu1b+V0sgZVVwFukiEe3z6DkHRhy71tfTYdfutiK8UM2QxNfIFe1wxrTAWRMHRpdKgB3qvf7
yy5Y1hchxTA2GkpOTxf79+3U1TK/nnntOpKamBl3X3MMDZcgH89OFQxHq4U+Hy9sRdlsmvxr0V1G1ePRp9SQrZM0Vqaqro1q2b6Nu3r5g9e7b49ttvwz5
XUVGRANDmZnQwFE1QoZbRf7tLStpe0LV8S7fy4lxSEv459F9I9Z4D6+tpdlBoiBBBX82ZJEj5I4lqUOKftDo0
ug70nc3Nt6j0jchAtwZDmnKFLLrkEBw4cMGqUTpwcMGqUTpWampqghR4zMjJQU1MT8ngGTJk3CmjVr8O672L58uXo4
di8bGxpCPuffee1+1FfX++/++VVsXIJS6MyD1yci6PGtGcAyOOXevr6YqFSkMUl2pMz8EUbMArCJ6MLIQ
N+U5KmfRQxWUlSc5UtrhMuleWGyEyldZIq7S0VPzkJz8RxcXFYtu2bWLXrl0BNy1C9C9UvG3dulUUFRWJPw/PzmjdjFhGf1jZgdn+Px4vzfOpjVmbZndW7PuPmdlPeYNZ8IFeGOgmxOiqiyAeBhcr2WgBk6+/7jVwRa+f+ePg2k0R9cl/WmVv9DuluG8ByN5q1AIJs4bS/zrrQ2Jj+Hn/f8eBQSr8E6NYiWKfw4/gj0NalsKWs1kgy6V+rIL6cZxpR/C0sP/c1/DZPlYCKxFcq2iYHeb9WdU6XWa/tAHGXbmzjh8U3B0yGXMhUuS9Rk2IRgVBO7PMdZ8Jkqxdg9m2rj3Tf2o0Gvw05Nd3/r99l2sC9OUvv/fmnAH8DlL0Je5/gTr4m+NjR6LeY2NbA2eNbyWCw1DhewaQN3ARfdvVdy47T4tdYtMaewyIDQr1BJd0WbDmvFBzqs94R3pDg8Ka8tEZHyz28Wd0LFqXCfD9rvlNUIZPRPb79Zf9p2XnwDDR2+nXxY9cZ3Q29oH2t+cfPycB+rm4HaHz5r0oiJEF8C3WmD5t+ikg1/ODmAkJKI4t94hK5VMnhzl1bIc6NxRq8bNGHnSiKmIi/8CbW24SRn9sTaTdtaFYRSbGdxqoFDcGTWuTW3fUNoNvVBv/KNGfwsaOzhS5Fzh3gyMMSZlQtAPN2aLKQyu3jX6HEfGnrRyXUOv82uRG8QMcSkeVvDB4hwzrg/pR87O6kQkzDcJ6kdT8ANpP+sLvVzzIK9KZkpl9A3l33iN3dOdA+Fpkv7JbAU3Bt7sWbOqQGEaX5KKTwaU81b3kYhP3YDeMtNGl5+wAlOwEHHAxfAUKe2TMzFcaUjKlG3s+wa2fRo7lazytAEKfO7grgvn08Ngc12pHyyDWuTr7wXIkBgp5GJC6GdtpnZXoPT+IWNH+cTvKjMfl7yj2jmGg1KxK9FKW9RktCIIFRRm2O1m7MZ85R0l8QCqLrZfrxL7FxlfPDp1WTL6sRfnHBBehuPGCICL4thUOpHRxZLsG/C/wzBFqWEvAGvxpMA1SKqxqcftO1WaeCWl5qzoZVHuutLOk7TVEHmBgtW3N2fqFdqGMIv32NF79pXfpnFY0AOVVOwiTxtbnkIlibCV1Y1J/TrXrjxzVJ3gnbVFU8Q5bsMO0GiLMx0xqj5lB0cs5Kf/OU/XB65N1ZkOCwAD23ThgOWsKbtUdWNi8h/4uZZ/N2eUAp1h6wW6PXYqvLG9I/X5bqgA+1DZJ3DWSpwAPgUWjQwJ4b2LzE2j0UjWlTzpYbXcAjuZmFukYa93TaPKpONavVS2gUwgnKqZcx+ynjOtnpg26XsOUG7ZNdULKWFrSwOeiZCY9/Au8bvrtSCiGKgbS4C9SMaAXbn9Ms/IpzkCimQrNxvAM1GzFbbYzN17rXJ5LDkwLqEibNyD4sYFGfmnXbFqFCUobxC/x3yQ1kqJM7p4HD/mwgU3HqOfQqFZAfyt9Dc6djhS4k9WqCfnUs9kAlyr/WQpC5m/TiXzNgHuxbEbnCiQ0Byc9WDGcYp/t/x/vNdHoCpdaT3hz3dt/7uMthmWr5Bl+9m2gxO8rBfeVgv/7BzJzE+kPgqxZ/e8RVdoTrYOLnZ3VsNcCkAfiZ0xSAKkQsLMnyCl0GlWgEoYBPvo4B5Yv7G+u/4eHGzHKxbg5LyQdeJhtfeY8grUlBezmWF4Xrc2pHAybqwXf3fPfzLWZSlq4BmJdQdYH3LAo7AAsmyF5jnoOYuHc/pOOEAg3yNhbX3IgHx34/XR7Ei7m6Z3Htky6Rx2+Z0KZ1oKbV7fGPBfkjf6dOaeAfNhdB3cFlGVWPmbwL4HBMjq+/mkaPmXjvIuqu+/Vi4b4OVSrBBjY7t1/yWkfoAx6bPDbXDm//Lu5H8KTt5hjR33Z7i9AFmcVvCV2+C6VA7wi9m2MmxqBA3KpVh2qDJYcRiB3fZQkMmNDkUbtmQTZt8TWbd3Nl7Vf/hn1wUk6MTlDHB/GWT3AWkGvYrIhflLAA+9ZNZqSazDqRq8xSgQNvBjdRW9qkLuqTdTzp+dTEsY58D6uXIa7nw5h09jwP3WsR+ZmEtLCBtePVC+XhiW6bFXyCnzoMVEK4hxBeYXWk6/QdHKqT9tmmuIYUeEuoxKoWXNYHGZtpuxDpCLcfWTl4hNHEZVHuN/aJx9rb1ht2YffkvF1yGKHOKl5B0c3e6Ap8Qh0fvYP6SAULvuFT5GpQCBNRtPswwADWsnGQDq/Z8GZZuKLmnZ/kZy1P5Rw9QcXiJR/FQnGJHxBVpsZvmAZbfs2dyDGPaxqOJMPVDjSLuyz9Sc6RgHarhaEKm5YKq0GNzfrnV2MyD8vQMUl5hYoyIDOOhpRPZXFl91jOGazLH2rnd1+wGpcdqSLE3Bhvm3wmuIS/qeydBPDIH7gKvmcBCLsG5xXWSnqcVazN6r8mfRzd0SEEVIxqRwFW49EaNSjR9JbCJ5gvoGhj89bbehZ5mXA5RAeP1gxTD58OLhAzFJ/eRSbNADzhOYZUUxdPsv3Pc7Hn0tD8BqAeCl9TvRpjcR1JQmHGBqCrrp+28rmi+XjcHlfrjp4a5EyFVwLfK47+B4VxZVAYJ2RRtxtuZI/LVQYnUBxFAZl1FP5gmnBVGyFiL/nBZwKMwT4o3GxsIKCxDDmwKBaxc2RFDvfmE7eGhV9WdDNqapm6dOd0XBBKz52Bq7I0+awi2rKOwNGbVwQGNvzKDQRoXJn10UCV2ucZMO7CXJtgC5FAFQFZYGk13BVzZCp8BHRxvcuJFp1dS5jQRd7LYGjPTqmQx9Mm6GMjjgm8a8T00jqT7tprLjqTzKoQbDtfB82KLZnlaY3ZO8PT3X3VA4CzL7I23v3+M/3PVeRjHL0ZzfcCtfnE9f3nuo24Sc0LN2/K+37QLo4QfDLJSwxbOn1zKaurF0NUJmWp7ypf9ojspSfg3/fsLNrb1O+AkNVPwwE9HOr9yVVGM4vKx7QqHAcL2MTAsDo3aa5QZrc03EWhF03i4/lhMS9L7ENKLOFYdwnXFaZ7uKCk58OCGOaAjTOM7u9S5NX5zYYtPOr4ka9AhAnnLqEkMTdPcTJtEVPHhHfjHgbnQaudhxkfTFpWgNpUmtTdhGwHSyrHKCl5yGzmKnxCg4F1xEUhjy6YJDqX8ztHFpdzEPWxOHJDZcnbpa0ZYo3vrx8KlvIxnOHKLDiF1ZQHhd2l6TEbaxNvWt9amSvETyVkRhlSyg4

```
+kCTJf2vXrp3/31jjpmBI65dEN4v2Qqv1GqKn103NY7p10x8Iae0J08uozgQnX0+M7HHU/MTRnBSnBQ1Gsu1FCdE
OOz+EpIupwdDAgQNFYWGh+Pjjj0VFRYU4fPhwwC3WuCkY8krPUDBarylaA0c959bIGkbK0iDRBCPRiLYzwQ3XE8t
HkIw6KWYGDXb3Itk9rOeF3IMYZmowlJycLPbv36+rYW7kpmAolr8khqPnmqI1uNHT66b2MWpu77zj/JGNUNx0PbH
s2m/0STEjaHBKV56dAZmXv2HGAC3Xb82zycaOHYtdu3ahT58+xmVxkyHcOJMnWmpnFk+eHHjcx45F3nfLWkF6Zla
pfUx6OlBXF/wYQk3tdxM3zZIONfvOcEafFDXTA7Vw0rRUy16UIDirzTM0B0NXX3017rjjDuzevRuDBg1CfHx8wO+
vueYawxpH2inF9iLVkIkVeq4pPh+wcGHkfa9YcfZaomdav9rHLF8O3HBD7AawvJ4EYcZJMSpo0PsNIxa5rVgl6aY
5GJo7dy4A4MEHH2zzO0mS4GNFUNsZ/SXRyfRcUyIFUIr09LP/19PrpvYxhYXyfWM1gOX1JAijT4qRi5m6qSvPbG6
sIkq6aA6GmpubzWgHqaDl752dPctW0nNN0fulXE+vm9rHmB3AGnmtVJZ6KS+Xfx4zJvwwHq8nQRh5UowuOc+uvLO
8mHvgVRbkMLmaUxKonZLL6DR6ksajzYnUk89pZw6oke+dkhK5GGXrcxWsQGXrxzlhlrSjGHFSzJim58SkYa/PaiN
dTJ1NJoQQ5eXl4mc/+5no3bu36NOnj7j66qvF+++/r2dXjueEYMgN05LtpPWa4qVZd0a+d0pKIl8fIwVEvJ60Es1
JMWuantYPiNmBilO+CdodkJFmpgZD//M//yPat28vrr/+evHEE0+IlStXiuuvv17Ex8eLNWvW6Gqwk9kdDLlpWrK
dtF5TvNBTYeR758wZIbKzIwdDrat2B9sPryet6D0pZvbgqP2AmB2o8JsgRcHUYGjAgAFixYoVbbYvX75cDBgwQOv
uHM/uYMiJPdZOpfWaEus9FUa+d7QUkOR70SJml5yP9AExO1DhN0GKkql1hg4dOoSrr766zfZrrrkG9913X1T5S9Q
WcxnV05o0Huuz7ox872h5f/G9aAA1Ge9mT9ML9wExcvp9qGPlrDaykOZgKDc3F5s2bWpTdHHTpk3Izc01rGEk47R
kc8XyrDsj3zta3l98L0ZJ7ewwK6bphfqAGBWohDvWxkZ1bWT0TQbQHAzdeeedmD9/Pnbu3ImRI0dCkiR8+OGHWL1
6NZ544gkz2uhpnJZMehn53hk1CsjOlvcVjiPfi0bWFTCblsrPdk77NqLbMdKxLl6s7jkYfZMR9IzDlZaWiksvvVR
07dpVdO3aVVx66aXi1Vdf1bMrx7M7Z0gIbyT7kjmMfO9EO5vMFk6ZiaSG3hwZO5LfjKhPEelYc3LkrH0vTPskU5g
+td5LnBAMCRH7yb5kHiPfO3rrDNnCbTORogkwrJ6mF219CrXHumQJvwmSblqu35IQwTrQQ9u6dSuam5txySWXBGz
/5JNPEBcXh2HDhhnWa+UEDQ0NSE1NRX19PVJSUmxti5t6+8lZ7KxAbQufD8jLC53XoowRVlQ4p+Hr1gHTp0e+39q
1wLRp5rcnEmWYCwg+RBduMVctx5qY2DavKDc3NtaqIVNpuX5rzhn69a9/jbvvvrtNMFRVVYVH30un3zyidZdkkq
xnOxL5jLyvRMXB1x+uXxzLDfORHLbbIloVoXWcqxjxsT2tE9yBM3B0J49e3DRRRRe12T5kyBDs2bPHkEYREUXFjTU
p3DhbQm99Cq3Hym+CZLJ2Wh+QmJiIb7/9ts326upvEGavCl5fLwzvl5fLPRlNmhwFnh5oUTl4UVAl
Upk1TP17q1mOlmKU5GBo3bhzuvfde1NfX+7edPHkS9913H8aNG2do44iIdFF6HlpfaBWSJOedqF0VPi8PKCiQ81w
KCuSfS0uNbLFMGXrKzg7cnpMTPgfHjbx0rOR4mhOoq6qqcNlll6Gurg5DhgwBAOzcuRPdu3fHxo0bY67wpMSqL2
ISeOkWzQJvq330frPpJZ96GH3G9/K57f7WClmabl+aw6GAOAf//gH1qxZg127diEpGtGnTEB8fr7vRTsV
gyD5qC/FScLzGIPibSO1MJDfOSDMCP3gUI0wPhryEwZA97PpCbhS7AxFez1rQ+2KUl8tDYpGUlcVOcq/bP3hELWi
5fmvOGbLLww8/jJEjRyI5ORmdO3dW9RghBgvXowePXogKSkJY8aMwRdffFGFuQylqkdaABOQ1II3OYTUqR9bKFJN
Qzz9lStsODWWVA6va4Rh6EnwBd85Ii4ZdHzwiB3BNMNTU1ISpU6fi1ltvtVf2Yxx57DCtWrMDvf/97bN26FZmZmRg
3bhxOnTplYkspWlpKxBjFqADG7kCE1zMDua3uT7Ts+OAROYRrgqElS5YBuU+KBBg1TdXwiBlStX4v7770dhYSH
OP/98/PnPf8bb333PtWvXhnJGml5WHrjXesKIWnBNMKTyqWgP7VyZEGGbaoRl8n5qq0lkG
FampqMH78eP+2xMBEjB49Gps3bw77mObmZjQ0NATciIjIRSI2UVXZbs8q5yp9rY5sf1Ybc23yIWLMJwQiaq9TmzZZ
rCSNqQXMw1KtXL9TV1bXZfvLkSfTq1cuQRhmhmLYGANC9e3eE9e/rA7d27d/f/Lppah5iyjQNLWd1JaVX8iNDGCc8MV
a7XXqoYfsyWfSyu5hR9Nr4YTq+bHjwJ3QE0ZkE83B0OHDh+EL8lWysbERERVmzZ2zatQwgtfp
gCyHabGspMTERKSkpAeylpVfyI0MYJzwxTrS9SwYpyZWO2HYEYAc8Bw+LM8aW7tW/Reil
Yze6eMCIbqV4/43//93/9/3/rrbeQmprq/9nn82HTpk3Iy8vTfAoZu5ehh3BCD3EGW1uPrU1ta
26S0i54lmDUgtjAxgnLC0lHI9mzJFfj41hTOEkO+7YIG8zJRTrnWOWmvV6LWxQk1hr6oCpk4N/9hQB25EPYVQH7z
0dOCmm4CuXeUAyylvEiKjCJUkSRKSJIl27dr5/6cEhISRL9+/cEhISRL9+/cTrr7+udne6e6FRcXi9TU1Ij3a25FpmZmeLRRx/
1b2tsbBSpqani6aefVv189189fX1AoCor6/Vr5aefVv189fX1AoCor6/X01yK0pkzQpSVCPWF2rfzvmTPG7z8nRwhJEkK+ygTeJEmI3Fz1z1tSIj+
m9f6UbSUlxrY/XDtycoIfU7hbWVmrHZn9AoSxdq26Nq9da1mTjKG86bS+OOEOXHnjBXsD63njKa/7ggVCdOsWuM+
cHOveyERR0HL9Vh0MKfLy8sSxY8d0NSwaX3/9tdixY4dYsmSJ6NixnxyoZb6Nixo9ixY4fYsWOHOHXqlP8+/fv3F6Wlpf6fH3n
kEZGamipKS0vF7t27xbRp00RWVpZoaGhrQ/bwMhmJfpABm/Xpt8UCwQCQ31/rrx5kzQrzzjhD/8R9C/PznOgKLYAd
i4YWwrExnAOd0ag9M7YFHCq60RvQKowMsK9kYxJNzmBoMBXPixAkjdhPWzJkzBYA2t7IwfkBiOLiYv/Pzc3Noqi
oSGRmZorExERx2WWXid27d2t+Xj1uRFEc4DH1uqn1gsh2GZ6sKmKfxY+5JoScsTDuaBy9AXxkQ9YcGHn+eEOuMM8F8K
Fm4N4cg5Tg6FHHnlEvPjii/6fp0yZIiRJEj169BA7d+7U+rSOx4AhPj45JcyYRUm9XTbOPUNqDtyM8US3dss5I7k6d
hF1gtPUNqDtyM8US3dss5IIgn5zA1GMrPzxcfffSREEKIt99+W2RmZop33nlHzHnyEBoz8V9aW7uLRKw2Lu3L/D7GRY
oHtBMa0pKm+uEwy6Etg87mpGPo6bLa/16dQduxuvl5IStUN+S3PyhJVOYGgx16NBHDlyRAghEDLyRAghxPz588
Qe/fuFZ07d9a6j08djMGQ9J3y5c1g8ECBSj5nWlJQ211cHXghtG3Y0Ox8nUs+PmgM3Yzwx2g+AWS9YuG9JTv7Qki1
MDYaysrL8PUP9+vUT69evF0II8dVXX4iuc41oOnTpp3Z3jMRiyll1f7lr/7X7hBfvjgWDXEzU9ZmpjmXnzQlyn3HJRsSJ
CMvNcGNnlZfR4otoAq7FR35tUj0jfkhYssP9DS45iajD061//WvTs2VNccccUVIi0tzTB+b68UXXXxRDhgzR3lqHYB
kLTuuw8H+dqen2xsPBGtTWlro61LL613U59BxmctBWDWOanYvWbiATmuwZ/R4YqQAK9jsArVvUq3UfEtqXQLAqUE
8WcbUYKipqUksW7ZMzJ8J8/X3z2Wf+7Y8//rhYtWV1t05HoMa1QM10diiKeEBPm1q2x5BYxjGZy2HaZvQFNxi
1keWSJcY9pxD6gz0jg6tQ7VCmWUbztVK7evQrZuzg3i1nDADdNQaYFgw1NTWJWbNmiYMHHOWhaM
NrYwHoq3Hp5wXQ2IZ2zOXg7B6HFXtC5KTY2xejNHBXjQ9aa0vzI2NxrxJtVD7LWnBAucG8WrZPXskhpjaM5Samsp
giExj5QiNli+bVsUD0dbja12UOOpYxmnfUO0YR12yxLrnNCPYMzq4MvJNavRzlpUZH8Rb+Rl+q16bTL
```

Ftddei1dffRULFy7UuwII2cznk5c1qq6W19waNSq6pYaM3F+4tbXsWqT18cflRcuNOl9GtCmUlmuoFRbKa41F9do
YvSZXtIxcWVetvn2te061C7I9+STQvXvkFzXSard6FqV77TV19wtFz0rFWhb+i4sz4I3/IyPWe1PL5wN+9StjXyt
ST2uk9dBDD4nOnTuL6667Tvzud78TTzzxRMAt1sRaz5DRPbBmThwxe4TGiZOm9H7pdlM6RFTseNGsfE491anDfeC
MbntJif4eoWjfpFbnsVndS2NlD6RHmL42Wahbfn6+rgY7WSwFQ0Z/ts3+W+G2RVqtaFOodgacb6cNbRnJjhfNiWO
3aj9wRs5IMGKBWTd8SxLCnty0rl2Ne61ICGHD2mSxLFaCIaM/27FS7NWJk6bCtQloO3s54FrgheRLO140q55TTzQ
c7gNnZM9QtLlCRs24syLYf+cd486bGlrOLXuGVGMwZKBYCYaM7i134hCTXk6cNBWuTSGvBUZ11bmhZ8mOF82M5wx
VWTNY4KXnA2dkr1a0C8y6pUejpMT6Xhq15zYtzZmfR4cyPIF64cKF+O1vf4tzzjknYuL0ihUrosphInMYnXdqRx6
rWQxJNLa4TW1ymo1KlLUyYTQadrxo0TxnsFkGr70W+lxv2ND2d2q0/sAZOSNBT+KzkY+3QmmpfK6CfY6CMeqY1O5
n/nwmT5tEVTC0Y8cOnD592v//UCTlw0WOo/azZtf97Oa0SVOAxjapnYX0wQehdxrqQlBVJW/fsMFZAZEdL5qe5ww
WYKalAXV1be/7zTdnz/Xhw2cDqG+/Be64I/JzBfvAFRYGD65ycuRASO1rGmlGVygtZ3o5WbgvFK0ZfUxqzm1aGnD
//cY8H7UhCaHuXX3o0CHk5+d7LuBpaGhAamoq6uvrkZKSYndzdPP5gLy8yDNTKyrUf9E1cn8UpXXrgOnTI99v7Vp
g2rS225UXNFRAxRdUvZa9QPv3A0VF2veRmxt4ro34wBlRA0MJmAH1QQPgvEA6mPJyoKBA3X0lyfhjinRuS0qcfw4
dRsv1u53anfbt2xfHjh3z/3zDDTfg22+/1d9KspTSWw6c/fuk0FO/x+j9UZSi7arT0rMUJZ9Pvu6sWyf/6/NFvUv
nKC2Vg5aCAjk41RMIAW3PtREfOKVXa9o0+V89H06llyk7O3B7bi7wm9/IAVlLOTnuCIQA9WP6aWnmHFO4c8tAyHx
qE5EkSRLffvut/+eOHTt6ohJ1rCRQK8xYy9FpyceeFG2irEWLwsX0ZDe9C92Fur3wQvDncMIHLlSSvRuS70NROyv
knXfMbYebz6HDaLl+qx4ma9euHWpqapCRkQEA6NSpE3bt2oVevXqZGKrZL1aGyVpycgVqikKobnY1QxVqhwjKynT
n6YRKSXLTSEpIkYYZ9Xj8cTnpPdhzKR+4H/8eo7bWPR8+p/7B4Nh/zNF0/VYbYbVr107U1tb6f+7YsaM4dOiQnmD
NVWKtZ4hinN6eA5MLC8ZKXaqQoq3Bo7ZnqCU3drNZ2WY9PSxOLDxGupmyNpkQArNmzUJiYiIA4IcffsDcuXNxzjn
nBNyvtLRUa/BGREbRO/3b5EXhjJjs5mhm1JBonTvSkttm/gHAyy8D11/fdrsZbdZbIsKomXfkOqqHyX75y1+q2mF
xcXFUDXKaWBwmIwop2EUkNzfqC0G0k90cb9Mm4IorjNtf69lkLblx5t+GDcCNN4bOljeyzUaMxzp1KI800XL9Vh0
MeRWDIfIcEy4EFqQk2SdYABmJUmcoVC+czfldhiotBa67Tt19o22zGwNFMo0pU+uJyANM+kas1JQLVaZMkuTOEKf
X5WtD6YUIFwhJknxbskTu+iork4solpS0HQpTMxXdTeXflUKGakXbZgtLRFBsUZ0zREQxzsSlOExOSbKH2orF2dn
Bz6He/C43lX+PFJy0Fm2b3RQokqOwZ4gcLaYL9DlJqB4OJbnVgIkRoWrKuakuXwC1F/rVq0MfnJ5CiG7qZtMSdBj
RZjcFiuQoDIbIsVoX8y0okH/mhEWDRVrkFZDr3RgQiRYWyktulZWdHTGqqHBhIASov9DX1hr7vGqqUc+eDaxfb/8
3CC1BhxFdg24KFMlRGAyRI1nQUUGQr5M7n7Q2z8LfGXK9D2NQjrj1Lu32s7MXIlQ3W9eu8q2oyNpvEKG6cCMFJ4D
8hli/3piImOsEkV7mljxyPxZdtF7MF+hzCKX+3Y2wZimOoE/upoKBrZlcqFJ1G5TCgkuWhG6HmQUDI72WoQoZKre
XX7amTa2Lj3LZi5in5frNYCgCBkPWU1vMt6zM7pa6V8tltEajzJgTrvbiEmoNLzdW+dVSsdjMi69d3yDUvpZ2rKk
W7nzHQjBOETEYMhCDIetZtGaoZ7W+brbDGXEEOcKHKHo41F5cYrHbT82F3uyLrx3fILS+lk7piYmlYJzC0nL9Zs4
QOQ4nhJir9SSoZsThdjzx4/915FloSfCKxTowkbLCrUiAs2NKudbXUs/MOaNZOFmA3IXBEDkOJ4SYK9j18BUUYgo
2oAoa571rvbjEah2YUBd6qy6+dnyDcONrGYvBOBmCwRA5DieEmCvU9fAVFCIPhzEGZZiGtdi57B2guBhobAw920v
rxcWN3X7RFLuy6uJrxzcIN76WbgzgyBIMhsiRYq5An4OEu242Iw7vS2PQKS0RF6ycJS8+Gm6KttaLi9u6/aItdmX
VxdeObxBuey0BdwZwZAkGQ+RYMVWgz0EiXTevFaV4pm4KJDU5LlovLm7q9jMi18fKi6/V3yDc9Foq3BjAkSW4an0
EXLWeQjFpTVPLBFuKrGeOD3v+mYfkOpWrfiurhFdVBc+LCbVKeLAnz82VL55OiHaNWv1c7/mJhpY3phFv4lCv5fL
lQLduzvuAKEEuEPiaKAESu55jhqbrt+lz21yOU+spmFgpU9JmtvM7ZdqnaGuptRP2yR00nd7IqeolJeH34aRaO+n
pQqxfH/3zvPyysz8gdtQ9IstpuX5z1XoijZQvlq2/6CujJ276YqlMgvJbpyPHRRmeCbbifbienjZP7iCxkGgbrMc
mJ+fs0FawN/Hx48D11wO/+Q3w2GPqn6vla1laKu/DiA+IWd2vhYXA5Mnu7tptye3d1E5gQXDmauwZopZisWZggGh
6RJzc06OVUT1DTq0MnZYW+djCLZMR6rU28nhjpfvVbDxPIbECtYEYDFFLMb9UiBPW23ICo86DEytDq7116xb8+MJ
dfI06XlaJVofnKSxWoCYySSyMnoTlxhlCZjDqPDixMrRax461rX8UaYbda6+p23e442WVaHXUnKfbbwc2bdJXI8t
jGAwRaeCJMiUs8iQz4jw4uTK01n2pufiuWaNuv+GOl1Wi1VFznr75JnKtMAIAMIGaSAOlTEmkmdKuL1NiRoKpG5M
8oz0PdrxhjAysWu5LzcX32DF5Ov3x4/qPN+a7Xw2i5/jdOMvDIgyGiDRQRk+mTJH/rrf8ex9zo0jRzvZqGfzs3w+
sWhV8ZpPT/yhHcx7seMOoCcC6dAH+/vfw+2ldfFDtxfemm+Rj1nu8nuh+NYCe4xdCfh0WLJCD/Jj4Q2UMDpMRacR
RJBVaL2NRVGTuqu1O5sTK0KtWydPnQ5EkYPZsYP36s7kmai++kydHd7ysEq1OpPMUCocZg7MgodsQDz30kBgxYoR
ISkoSqampqh4zc+ZMASDgdskll2h6Xs4mo1BiaSa5oULNcPHy7DQhrH/DqCks+PLL8qyxlvdJS2s79T4n52whRbU
z7KI5Xr2FPL0m1HlSc1u71u7Wm07L9ds1y3EUFRWhc+fO+Oabb/Dcc8/h5MmTER8za9YsfPvttyguLvZvS0hIQNe
uXVU/L5fjINIg0jIWoZSVObcAo5upydNqPZxZVNR2P0rvw113AcuWyf8PNgRmZE+X05dscYpg50kND3zmtFy/XZM
ztGTJEgDA6tWrNT0uMTERmZmZqu/f2NiIxsZG/88NDQ2ano/I0/RO6/Z6MqxZ1OQ7KfdRAtlglFyTF18EXnoJWLh
QW7VxPWKtSrRZWp+njAxg1iwPzPIwlmuCIb3Ky8uRkZGBzp07Y/To0Xj44YeRkZER8v5Lly71B15EpJHeoEZNPoo
bZ6O5SXm5uint3boBhw9b81o4eckWJ2l9njwzy8M4MZ1APWnSJKxZswbvvvsuli9fjq1bt2Ls2LEBPT+t3Xvvvai
vr/ffKisrLWwxkctpneGiNhm2dUI2a6Zo4/PJwU6o4nvKemJqVFefvfhOmyb/ywurs3CWh3amZzCFUVRU1CbBufV
t69atAY8pLi5WnUDd2tGjR0V8fLwo0ZB8xwRqIg0iLWOhJxmWSw5EJ9LaVVoS3tUspUHO4fFZHq5ZtX7evHm48cY
bw94nL9QYtg5ZWVno2bMn9u/fb9g+iaiFcHV1WlOTZxKp6jFrpoSnLJ/R+vwpZQ3WrwfuuCP866Rgron7cJhRNVu
DofT0dKSnp1v2fHV1daisrESW14t1EZlJ6aJvPcMlJweYMwfo21d9nomWpRn4Rz+QmkDyttvkqtFqRZNrwpwvcjD
XJFAfOXIEf//733HkyBH4+fnhueeeQ8eOHfHee+/hzTffxIcffoiuXbvi+eefx+jRo/Hxxx/jzTffxIcffoiu
XbvihRdeQOfOnfHee+/hzTffxIcffoiuXbvihRdeQOfOnfHee+/hzTffxIcffoiuXbvihRdeQOfOnfHee+/hzTff

fvxHu3bsX9fX1AIC4uDjs3r0bzz//PE6ePImsrCwUFBTgpZdeQqdOnSxvP5HnGNFFz6UZ9DMyQFy/Hrj8cn2PjTR
Ux4RecgDXFF20C4suEtlIqX0TqWZKRQWHXForL5dn3UWSng7U1ZlzfiMV4eTrRybScv2O6an1RORyatbZYs0UWev
p8yNHqlvj66mnzv7c+vfA2fMbaXp+MFpyvohsxGCIiJyNNVMiC1aHqXdvuQ4QED7QmTo18vnVW+eJOV/kEhwmi4D
DZEQOwdlIwYXKyWm5nti6dZHX+Ap1fiPtP1xAqnaozgPrZJH1tFy/GQxFwGCIyGQMcvRTm5Nz4ACwebP2cxxtzg9
zvkhhw+c8JhdqJaIYxCnX0VGbk7N5s76el2jrPIUrwsmcL+9wweecOUNEZA9l+KX1xVaZcm30umN6EoCdTktOjp7
jNyLnhzlf3mb151wn9gwRkfWsXmbDBd9MdVFbX2n//rbDXWqO36g6T0YV4SR3cdFyOswZioA5Q0QmsDKxNpoEYKd
Tk5PTtSvw97/rO37m/FA0bE6gZ50hInI2q6ZcR/pmCsjfTN06ZBapDpNyjHqP34g6T7E4PEnquKi0AoMhIrKeVct
seKHoX7icnCVL5OrSoag5/mhyfvTWJ6LY4KLldBgMEZH1Ro1qe3FtSamOPGpUdM/jom+mUSksBA4flocb1q6V/62
oAPr2Vff4SMcfav+RAiEXJM6SiUaNUlcFPdrPuQGYQE1E1nvtNeCHH4L/zsgp1y76Zhq1YAvjGnn8WhbedVHiLJn
IRaUV2DNERNZSegxCDd907WpcUrOLvpmawq7j98LwJKnjktIK7BkiIuuE6zFQJCXPJPQZGsPubqd3VtcMdPyD/PHu
28c/rleFJUidUaQVATqp3QLkF9gwRkXUi9RgA8u+N7DGw65upU5KHQx2/oqjI+HZ5aXiS1FGGWadNk/997TVnfD5
+xGCIiKxjR4+BzycPvT3yCPD448ALL6hLAI6G05KHlQToJuC/97odnl9eJLCc9rnAwyGiMhKVvcYtOyd+dd/Be6
4A7jnHrkIoZlDY06tbbRqVfDtRrfLiPpEFJsc+vlgMERE1rGyx8Cub59OTR62iIiI7
ZwPr10VUrtvPbp1OTh+1ol576RBTbHPr54GwyIrKW0mPQeuHUrl3lf4uKzm7Tu5iqlm+fRq+J5NTkYbvapaU+EcU
+h34+2DNERNZr3WOwZImcx9O69pDeIS07v306NXnYqe0ib3Ho+5DBEBHZQ+kxuP56Oi7O7IdZsB8ojlPH1HAgtzyOB4Ps7xmp6ocGWFU7/Bjw7I2m9
x6PuQwRAR2cuMhEq7v306NXnYqe0ib3Hg+1ASIlwpWGpoaEBqairq6+uRkpJid3OIYs+6dXLRtUjWrpULnUtqmlzCY
DgleetuKPrt0VqNW0KyND3lZb66w2Uuwz+fOh5frNBGoispdZQ1qhErVzcuRueCu+fTo1eVhpV2kpMGtW2/OjJ2m
dSCsHfT7YMxQBe4aITObzyYURq6qC5w1JknyBrqjQ963Rqb0zdlN6zlqfcyt7zohMpOX6zWAoAgZDRBZwwpCWlyg
BaKhcrWgDUCIH0HL9ZgI1EdnPgQmVMc1JVYB9PrnA5rp10RXaJIoCc4aIyBkKC4HJkzmkZQWnVAEuLQ2e08WcJbI
YgyEicg4HJVTGNCdUAQ6Vs6QU2mSPIFmIOUMRMGeIiBxNT4K42UnratrMnCUyGXOGiIi8oLRUDioKCuRaTQUF8s+
Rli+xuwqwk3KWiMBgiIjInZRhptZBhdr13OxMWndKzhLRj5gzRETkNj6fnHgcaj03SZLXcjn5s8OXzvjl1J607IWSJ
qgcEQEVEkWvJyrCjyqGWYKVJCuh1J68racZFylixeuZy8i8NkREThaMnL0ZvDo5Xbh5nszlkiaoXBEBFRKFrycqL
N4dEiFoaZWGiTHIRT6yPg1HqKaVy3KzQt078Ba6eK2z013kh8D5JJuGo9EUXG6r/haZ3+bVQOjxrKMNOUKXLgE2w
9N7cMM7HQJjkAh8mIvMjKIR230pKXY0cOD4eZiAzDniEirzFqWnasMyMvx+gcHq7nRmQIBkNEXmPktOxYpnX6t11
TxTnMRBQ1DpMReY3bp2VbRcv0b04VJ3I1BkNEXhML07KtoiUvhzk8RK7FqfURcGo9xZxYmpZtFadVoCaiiGJu1fr
Dhw/j5ptvRn5+PpKSktC7d28UFRWhqakp7OOEFi8eDF69OiBpKQkjBkzBl988YVFrSZyKA7PaKfk5UybJv/LPgvkE
ouS8ROYIrgqGvvvoKzc3NeOaZZ/DFF1/g8cxs+8qo7rsv7OMee+wx/PrXv8aCBQuwdu1ajBgxApMjRGiMMUEEOV2
FLSdyKA7pEBH5uXaY7D//8z/xmimk
c0iGiGOWJCtT19fXo2rVryN9gpqYG48eP929LTtHNGjsXnz5pDBUGNjIxobG/3/rq2tBSC/EEQsDlRYCA0NjbmD+
Mk7V28OBBPPnnk9T/tu8Fs3TpUqmIvpvubm5jSaiIiHMnWYGjx4sJCnsbdu
2bQGPOXr0KZOnIipU6di+9ZaGmUQbFtyHB+SWyIoWy1KzQmpwBZQhtWRmeWq2
/QWtZSYmIjExEQVrScivVvz/qjIqIyGJxEQVrSciIqJYYGswWkdvsZ
qQkFBAYyYOHri4mkK0axeZuHEjBjk5uTz
i2bJlOHbsGrsGGpatrk/gwYMAC73c/sJCVmrx3jzbdFe73+TCn
22zhw4AAOHDiDi+++eNf4bYT1J/h+PTR2zi/iUG3Z0UyWLUi
IKqwzRERE5D4xtxwERERHkVkYDBEzERkZGo1lI4IQ8VRdJC
iF/P5gA8+AKqrgawsYR+TEYIi85SfTE5ZR8tlcLdMRIuLNr
hAKiqkreXltrTLJWGAwREzLt9l5mjb+t6hn3YPGw9PRs5y
ej8hEDIaIiMHoWVnG3SlCvgcgZ9wRBPTfpdkVJmp2lJb6ZkU0ydpX9
DIuXnlStZb4gcgcEQEREFKi0F8vKivVMJF4hA1tR8dA/ZjlOhUv
9fT1SUlLsbg4REbnApqeQHwgt4HbJhaJTfb
jbSDyKAZDREQkYQ2g8igGQ0RERGNRmsTlPIPBWIpjcrRrHt
kyawNRJ7BYIiIqNqKiwPGjLDAZERERERERerqDISIivI0Bk
wRERERJ7GYIiIiIg8jcEQERERRorUEcghAAANDQ02NwS
6hdTU1LD3kYSakMnDmpubcfToUXTq1Cbm4uKisrkZKZ
4deoUevTogXbtwcFsWcognbt2iEnJwcAIEkSACAlJcXzbQeGIqIig
8jcEQERERERRqDIQ0SExNRVFSEbnVFSExMREurb
PYzBERERERERERNSZgiIiIiLGIuIiIyDyNwVAIhw8fxqUfnuqIdTrpEwJe2yg

NFU8//bQVTbRccXGxSE1NVXVfr71v1J4br7xv9uzZIwCIjz/+2L9ty5YtAoD46quvQj4uVt83F198sZg7d27AtgE
DBoh77rkn6P3vvvtuMWDAgIBtt9xyixg+fLhpbbSL1nNTVlYmAIgTJ05Y0Dr3Ys+QBvX19ejatWvE+5WXlyMjIwP
9+vXDnDlzUFtba0Hr7BXp3FRUVKCmpgbjx4/3b0tMTMTo0aOxefNmK5roeF5830TilffNli1bkJqaiksuucS/bfj
w4UhNTY14nLH2vmlqasL27dsDXnMAGD9+fNjXfMuWLR32nzBhArZt24bm5mbT2uoEQl6vz370rFzU8yTd2SuS4cRxBhArKwuXX345ysr
KzGymKzEYUungwYN48sknMXfu3LD3D3mzRpRpEtasWYN330Xy5cvx9atWzF27Fg0NjZa1FLrqTk3NTU1AIDu3bsHbO/
evbv/d17mxfeNGl9MGlI5539TU1CAjI6PzaFh79GWtqaoLe/8yZMzh+/LhpbbWannOTlZWFZ59
9FiUlJSgtLUX//v1x+eWX4/3337ed9aeoNNGl5539TU1CAjI6PN9oyMjLDHGYvvm+PHj8Pn82hZ6/zfccAOuuuo
qnH/++bj66qvf//3f9i3bx/eeOMNMw/LEGafNmK5roeF5830TilffNli1bkJqaiksuucS/bfj
w4UhNTY14nLH2vmlqasL27dsDXnMAGD9+fMhzsWXLljb3nzBhArZt24bTp0+b1lar6Tk3iiFDhiArKwuXX345ysr
KzGymKzEYUungwYN48sknMXfu3LD3mzRpEtasWYN330Xy5cvx9atWzF27Fg0NjZa1FLrqTk3NTU1AIDu3bsHbO/
evbv/d17mxfeNGl5539TU1CAjI6PN9oyMjLDHGYvvm+PHj8Pn82l6zWtqaoLe/8yZMzh+/LhpbbWannOTlZWFZ59
9FiUlJSgtLUX//v1x+eWX4/3337ed9aeoNNGl5539TU1CAjI6PN9oyMjLDHGYvvm+PHj8Pn82l6zWtqaoLe/8yZMzh+/LhpbbWannOTlZWFZ59

```
hIiIi8jQGQ0RERORp/x+rHFhG3rlcFwAAAABJRU5ErkJggg==",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plot_scatter(trials_logvar[1], trials_logvar[2])\n"
]
},
{
"cell_type": "code",
"execution_count": 36,
"metadata": {},
"outputs": [],
"source": [
"# Percentage of trials to use for training (50-50 split here)\n",
"train_percentage = 0.5 \n",
"\n",
"# Calculate the number of trials for each class the above percentage boils down to\n",
"ntrain_r = int(trials_filt[1].shape[2] * train_percentage)\n",
"ntrain_f = int(trials_filt[2].shape[2] * train_percentage)\n",
"ntest_r = trials_filt[1].shape[2] - ntrain_r\n",
"ntest_f = trials_filt[2].shape[2] - ntrain_f\n",
"\n",
"# Splitting the frequency filtered signal into a train and test set\n",
"train = {1: trials_filt[1][:,:,:ntrain_r],\n",
"         2: trials_filt[2][:,:,:ntrain_f]}\n",
"\n",
"test = {1: trials_filt[1][:,:,ntrain_r:],\n",
"        2: trials_filt[2][:,:,ntrain_f:]}\n",
"\n",
"# Train the CSP on the training set only\n",
"W = csp(train[1], train[2])\n",
"\n",
"# Apply the CSP on both the training and test set\n",
"train[1] = apply_mix(W, train[1])\n",
"train[2] = apply_mix(W, train[2])\n",
"test[1] = apply_mix(W, test[1])\n",
"test[2] = apply_mix(W, test[2])\n",
"\n",
"# Select only the first and last components for classification\n",
"comp = np.array([0,-1])\n",
"train[1] = train[1][comp,:,:]\n",
"train[2] = train[2][comp,:,:]\n",
"test[1] = test[1][comp,:,:]\n",
"test[2] = test[2][comp,:,:]\n",
"\n",
"# Calculate the log-var\n",
"train[1] = logvar(train[1])\n",
"train[2] = logvar(train[2])\n",
```

```
"test[1] = logvar(test[1])\n",
"test[2] = logvar(test[2])"
]
},
{
"cell_type": "code",
"execution_count": 37,
"metadata": {},
"outputs": [],
"source": [
"def train_lda(class1, class2):\n",
"    '''\n",
"    Trains the LDA algorithm.\n",
"    arguments:\n",
"        class1 - An array (observations x features) for class 1\n",
"        class2 - An array (observations x features) for class 2\n",
"    returns:\n",
"        The projection matrix W\n",
"        The offset b\n",
"    '''\n",
"    nclasses = 2\n",
"    \n",
"    nclass1 = class1.shape[0]\n",
"    nclass2 = class2.shape[0]\n",
"    \n",
"    # Class priors: in this case, we have an equal number of training\n",
"    # examples for each class, so both priors are 0.5\n",
"    prior1 = nclass1 / float(nclass1 + nclass2)\n",
"    prior2 = nclass2 / float(nclass1 + nclass1)\n",
"   \n",
"    mean1 = np.mean(class1, axis=0)\n",
"    mean2 = np.mean(class2, axis=0)\n",
"    \n",
"    class1_centered = class1 - mean1\n",
"    class2_centered = class2 - mean2\n",
"    \n",
"    # Calculate the covariance between the features\n",
"    cov1 = class1_centered.T.dot(class1_centered) / (nclass1 - nclasses)\n",
"    cov2 = class2_centered.T.dot(class2_centered) / (nclass2 - nclasses)\n",
"   \n",
"    W = (mean2 - mean1).dot(np.linalg.pinv(prior1*cov1 + prior2*cov2))\n",
"    b = (prior1*mean1 + prior2*mean2).dot(W)\n",
"    \n",
"    return (W,b)\n",
"\n",
"def apply_lda(test, W, b):\n",
"    '''\n",
"    Applies a previously trained LDA to new data.\n",
"    arguments:\n",
"        test - An array (features x trials) containing the data\n",
"        W    - The project matrix W as calculated by train_lda()\n",
"        b    - The offsets b as calculated by train_lda()\n",
"    returns:\n",
"        A list containing a classlabel for each trial\n",
```

```
"    '''\n",
"    ntrials = test.shape[1]\n",
"    \n",
"    prediction = []\n",
"    for i in range(ntrials):\n",
"        # The line below is a generalization for:\n",
"        # result = W[0] * test[0,i] + W[1] * test[1,i] - b\n",
"        result = W.dot(test[:,i]) - b\n",
"        if result <= 0:\n",
"            prediction.append(1)\n",
"        else:\n",
"            prediction.append(2)\n",
"    \n",
"    return np.array(prediction)"
]
},
{
"cell_type": "code",
"execution_count": 38,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"W: [ 5.52963938 -5.31347949]\n",
"b: -0.38024720917511434\n"
]
}
],
"source": [
"W,b = train_lda(train[1].T, train[2].T)\n",
"\n",
"print('W:', W)\n",
"print('b:', b)"
]
},
{
"cell_type": "code",
"execution_count": 39,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"(-2.2, 1.0)"
]
},
"execution_count": 39,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
```

"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkcAAAG2CAYAAAB1ZSLWAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABm9UlEQVR
4nO3deVhUZfsH8O+IMoAKLiiLoLjkkntaKmZKi1uZ5ZJpIe6i+VNEzaxU3HLLpVxySUXNrRLbtNLXQM0ll8Qtck
BBBEX0FSQ4fz+mGZime3MdubMfD/XNRfNmTMz9wy+77l5nvu5H4UgCAKIiIiICABQSuoAiIiIiBwJkyMiIiIqhMkR
cEREREERXC5IiIiIoECZHREREREIUwOSIiIiIqhMkREREREUSFMjoiIiIgKYXJEREREREVAiTIyIiIqJCZJUc7d+
6dUNgYCAUCgW+++/fWRoAQ8PD9SqVQsrVqywfWfRoAQkW7JKjv755x9Fo2PfWrRoAQ8PD9SqVQsrVqywfWfaBEREQkW7JKjv755x80bdoUS5cuNen85ORkdO3
atcPJkyfx4YcfYvTo0di+fbuNIyUiIiK5Ush141mFQoEoEdO0HvvOxIkT8cMPPPyApKUl7LDIyEqdOncLhw4f
tECURERHJTWmpA7Clw3CjGSdWfQrbCzMQ2hdqpkyZZSwaObbUeOyIRTMuzGqkKNnDUlLWVVVnAFL/
vUKdOHbi5uYl6HhERkTlSqFWrVqScUPEkrwyc7w4cPRq1cvnmSpPyuQGbCgnrLmbNcnPYpUrQ6FQ2DxmIi
ispwgCLh//z4CAwNRqpS4To4aySmRtqWHhlF+3M5Jifnx/y8/c//8OrpGgdOdOHnfx8pWeyaRRvn2wggBoCUovPO
/bvE0qD5l0z+Pi8ehfPHvtmSmi2c/LN+cRl3zfdaLK3lTSKDu4sUTBd+cxMrkwwW9PdtdDpn/FsrQ4Axe/EDwqGc
Oi+bJ12Glq5VcOsi2SsJxZYdXIHGM7ihhdMoEVIMLqZzwE1MtPB9ITBb6jRXmw5ztYl20UWfTnMyR2mQ4YAlwjqVs
JifXy+F6nzHYTwZnx1i1RqPNb4avjPdW4vGg7GaUETyL/EwPzzpvdrkRlXnhsjEy8IXfWCkVzjJbvbFNpsqnqLbVna
sZGJIwhhu7Tv7OjhLE5R1HVuVKYdH/BRkrC6GEYxPaGvqI4k0mWr4qlH5WKM79PoRDhXnRzGfHhPQDRd/eu7mc/9
6/TQKIWfkvtPmftYm8qsJ/lXcB+8zmixFhExQ0WeIgfiDLcTaVNXRM8zfVf1wYv4JsnM9AuV5/Ku7ltCNDwkv36Eeq
eDxQmp1+Pw5I9yMwQlOn/Ik7yDmdTqy6cGRJ8ifBhDJrpYCpH2Na0lbrUKg6Qb1EQJ1lH1PzbGTYd4AzO/XzZ/rD6
EYBTYC33GlpvpoPN5OJKUe38OTXV9X+1S2gXkv6v9krbRQi91fYDq7b5XIpyjz4oRJhWwbgE6xLAu7qlqbdIS1m3IIOt
jCzhxi1s1Ee5p/6+qGiRwE8LgCK6AFgBdnk5mgwXqFpoSsm/uo8KuAmC+XybBC5FvlonIKCS9XM+5yZ9dsTNk+jJbV
bE/4yIRLEtdGcIc7aCdn7yJFTqpzvPNxNqYWmJqvU/g9t75/4htm9u6yBdbsm9xtKIgP8DKLxpnUuaJqqmvrGQ6
CESnISGeqQZYMyknzcwCZ1WgUMDJeMgBe/9M0GeMHkbeUgSWTt9k7k/QoxjiLXF5E3DA+iBtH9Hwd3mcdE3fBOwlUmhq
OaldmH2E+7/Q17Vfgx4sdehhwvc+KoixzkAR2xuQXy8zAzKB0dwKJOO5IyAYEbk2gtqSMCpUrvzPcAuRC4LG2OgAF
Cs5oiv5OMgXd6aN/wsnmuWxZ1E/BL7SZkbSCmW+i8C3qHvhhEW4+Spq7NTaGCRDnpEw3tS1jjnHHKhiRHgvr1pP

atWtj/vz5EkdkfUyOHIGbm0P+2ZORkYEpU6YgIyMDAQEB6N27N2bNmiV1WETkIhx83YrLKlOmDMqUKQOVSoUNGzY
YXPksV5xWM8Lm02oEgN8lEZWkUqlXpaWl6a47UijU61qSkx1isN2lnDp1CgcPHiyyjN/RcFqNiIicjmbdSq9e6kS
ocILkAOtWXFrTpk3RtGlTqcOwGa5WswIOvlmO3yER6dKjB/Dtt0C1akWPBwWpjzvYuhWn9fjxY6lDsCsmRxbQdIZ
2ps32pKL5Dgt32yYiAtQJ0NWr6lVpmzerfyYnMzGyl/v376NJkyaYPHky8vLypA7HLjitZgE3NzdUqFBBu4+Yl5e
XqAaJpB4xevjwITIzM1GhQgW4cXyciHRw0HUrLmH8+PG4ePEiZs6ciVu3bmHFihVSh2RzTI4spNltnhutWqZChQr
a75KIiBzDrl27sGrVKgDq/TgnTJggcUT2IbvkaPny5yZg/fz7S09PRsGFDLF68GO30bLGRkJCAMB0dxJKSklC/fn2
rxKNQKBAQEICqVaviyZMnVnl1OmTBmOGBEROZjbt29j8ODB2vsLFy5E5E7dq1JYzIfmSVHG3btg1RUVFYvnw52rZ
ti5UrV6JLly7466+/UL16foL+/UL16db3PO3/+fJFlfFWqVLF6bG5ubAEEEAxGRUxAEASNGjEBGRgYAoEuXLhg6dKjEUdmPrAq
yFy5ciMGDB2PIkCFo0KABFi9ejOzZQu++eYbAEClSpWwZs0al6qplU1
ylJeXhxMnTqBjx45Fjnfs2BGHDh0y+NzmzZsjICAAL730Eu4PLj4w2em5ubi5ycnCI3IiIiV3H9+nW899572vtffPE
FAlysDblskqOsrCyoVCr4+fkVOe7n56cd9isuAqq1atwvbt2xEXF4d69e9erhpZdewv79+Hj46O9BQc
HW/VzEBEROSpBEDB48GDcu3cPANC3b1+89dZb0gYlAVnHVHAEoMawnCILeob569eqqhPvoPv30U7z
wwgs6nzNp0iRER0dr7+fk5LDBIiIissGglo6ACqmXxVqgjkKqdLwDZ4ImPHL8tlRbW0q4qrsyB2dcEi9fk5DBBIiIil5CVlaUdcAgMDTSpUsljkgaskmOfH1yDzPKGKDWHKeHFdBNiu16pX3nLX0JIF4NDLZW5rUAGOvDeWrZlt9M3
edLm5UyIiIlMplUrMnj0bnTp1kjoUychm5AgAoqOjERn4zW0K8CUsXBfPdtlhaUGJiX5NmzZYtWoVkFE9YQb3/A2AnJ3DQKm
vXoyQkBA0bNgQeXl5+Oqrr7B9+3bLNUwTeUoE/xLgiWSVHffr0we3btzF9+nSkp6ejUaNG+L3Ey6DxemjiOZNhvpOXKNUAGgvU4bZ5v9
ASE9PR0pKivb8vLw8jQPERE5FBQOnjyJ4cOHY82cNWjcuLhU4TgEhcDt0A3
KycmBj48/PsrOzizSSJCIkrvHjx/2WefxdmzZZ+Hu7o49eE/boXbAkN5Zcv2VTc0RERETWNWXKFJw9eeR94wYA0KBBA7R
u3VriiBwDkyMiIIIXd0ODAAXz66acA1IueNm7cd3d4mjcgxMjoiIiFzM/fv3EREZAJ/fv3ERERAU1lzcyZM1lvVAiTIyIhc
zbtw4JCcnAwCef/f/75Is2PickRERGRS9m5Wr14NAChdhtLtYbsxS+9m5cydWr14NAChbtizWZr1/PDdmLYXJERETkIrKysSB48PGDt/UWLFqWrVo
SRuSYmBwRERG5iN9++U23bt0CALi0ls4ePAgWdrduj1Xs5IIiIjkDvG/+b48BAsBvOvw3uIiSkEzO9BSz1nOZMqx1
r1q1x6NAhJkYGcOSIiIjIxTAxMozJERERERG4kd1sfWHpYT3PDOpE0//jpp3/wv//
9D0qlUuqwwHB5HjoiIiJxQfn4+JhDjoy3uMhyyEqb8wqAhiPIiGpM24p4Jlw4qtVmtsRJqsKKCeQ1ZOTsiAnfrYYFSRPO46wSqE4z+fuN/SeZRqsKQUq2HiA9Qc9Nx54PGSgJAWkWwq45k0/FOPDJ2+JGhIM+3Hb7p/e2s
ETubkyZOIiYkBAJQqVQVobN25E2bJlpQ1KRpgcEREREREREROZHHjx8jPWc+fn5AIBJkyahdevWVEklL0yOiIiIlckdjEnkt0wgxJyaHRdRrFvUFEIinmtMkjkyZN
x7tw5AECzZs0wZcoUiSOSyZHRERERETmL//v1YsGAABAMDd3R0bN26Eu7u7xFHJD5MMjIiIiJzF79mwIggAmDVrFho3bixxVPLE5IiIiIhcwJEjR7B161YAQKtWrTB+/HiJ
1aiRxRPLEPkdEReROYv27Zg0aRIAcExMxBYlJyuBITUVQkQRpTiJGbdp8RaYePNpv4hcSRhYn6xYxTqcW0zsCQAp1Q2I4I2ANERRK8dGb54ycm92K3a7kfbMbD0HjFzcX+Wc+BpBZZjZKV5bGHGFXT4pl6QIMjkUQ
vX17iiJwHV6sRkaypVMCBA0B6OhAQALRrB7i5SRy0rOlHbvA1S3/ayFZ0SXWRRJkIs0Yc3CBwdD45Q0AgSEwvEAsihtTM1bwDCU9UJi6jI1I71nNcB7iyH9U3Z8Y1p7aR8i3NpS3V7jQpn5ZhpAVoEGRGKhwHmSsGNUMugGJOvJYx0zPRf14+e7pk3WRjp95QmnAlsIgSJaXyH
EAKkGEOnd6vCV7/KpPK0MLWdZcLcnxc8tLBfD7TUJTvd5h8NZJLp5fZ/DE
QEqBBRZSmZ5rEJ7nYYU6u8MtfGHc4fzgaIODVrOd9zYrLbU5TrfccecQHEZYVQvo5k9bYFeMZbcZvSchtD0/Atq8Z2cVrKMcRMHOhc
OfKJWq1nMS5xR3Plm4+ss9ObEzBvd/QfcwDYWxrFdeS3Q9kq7TN2DV+tyzgFqqTxRiTZK8ZN7OUfFbBhGiYpm/zg36F+X5Xb5YFNYXc+SHrMSyhMMwAEUQhAB5P76o3hcJSPtR8W/+ghdrhGUUlFuxXHTtPYTM/xggsjiuVw8ExQ4v2SZcCVBS/9SVIIhgkzbFCCBwqcqMgsfcc4tTzZ7rFxG2KtqKwSwQnQr+Z2LPpzI9/BgAqwkFYGWeQ4MM5i7UhTrpydIQZwQwMhR/HT/+M/GvxDXScGzjVN/vu7cvH/v1fL9VFw6pfMq4+PWuo4nrf7pxtC5xkgkPlP+Y6+YV7JOfeScuvszoM1cVBcUmN8AoKxOpD7YEASfz3vVWcoAQ1gFgw1B2Pyv+oHD2+fUYS4ZUEiIzdTJUyORuzRbHxa+xB0Z8M+A+v/Lz90yPnjeKvDxYZ7dA+dxV9+U0Q9W+AVE8LVOmMdwMxqCY4wHqwk0v37Sm3Wfwo0hGU4YcF4vpvKDsb4J7GUo+wXfYNF7NQxrt1Zlr8ECAyKUvSWUw8mVCM0T9pyLVqmb1WgmNOjZVcmnCoUpfNbmy++RvrzKQVgaT53yyWGtvcIaAZzW/uN9Asq9pdMrPEeBYCPE8Y6XnHn1z8F7oL74nMfr39BeZHZnfqI1o/7qBcd+SXLNmS0fbOqyYxRhyeVuqTgmUDNw9CuY2YmrWr0CZCTY8B6AqlbLd1eFYH8l/PPYMOHuxOzmKeYqvwdCXnZyE7pqU6u/MW0TtTCSJQTBYlXYNHzJCBoNhHkNwlZx2/MvDAZ4rnnk5tLKkSMp1CY+tOYDTqlCLftEl8u6zvwRgVLjsqR2eOm3HYsltNYh2enMlm26Xar9Kr+MjvHBv5dJqQ4LSDjC2U+uFG/aQVC1+oaX5hKB9LV93rMU1jtiIYM2kKjuB8DAJ5/fixS2n4Z4wnbRn5ZCv6KxFPwMdPNsLkVe4IFn3qTB5pj7nb5ZIsqRdp+PwjJXGFYK+zCK7x67ywE6c/4aPMhsCqzx2vESHcnXmSZ+K5dqI8CiF+VJAqD4KcROnJ4rdW8m0tdXhyOEk9iUcA/hNTj1cO/yV0i8ly6LO/Xt6IhKVVdjGz4AAtXl1dmpRe+i4LX9wsP0o6PQmDu+eQ/j5j7iIaFR67Y/9+k9tTVHvzjL/x6ZPMeFn9yGmIRz8jlzsm2SgPxKAv9IyX8rmoKgl+FJZJUyUeh5W6iKkCMOw4kO9NqYQCZYt5NxRF9Gu7ayZJL7dQ8W7vAGtGhHdSOTROAxzNcyp7SJz5Sh4KpW8lFs6r2GHJXYxTAd3SXI6YhpbN4lPEq/Y1CldZ6nCDDkkSZkGHGGGBoR4dnvB0TPaoDYAtxL8aIKJV6vQDBCKHhwNmwS30nP+kgsfHLXaSc1vgAvN+bzcROs0QTHZTrAF0OwE0cXmHm3Ycmq7UFIlMAi6gbu9H4jFlh9tnPw30FzmXtMG/rFBfUOBDdVQx4UZE1JtuaRp/iNpjWBzOx2Y2wKAVCUpaFhFaSpVJDWxr5TVs3GjGlDjMNYqavlxMmGAH6cvzGmsWBPwjeSNLHwLKjeM+dz+t1rfkg3mJ9pLKgcQ0K11CZCFLaYBK4rWmCAfGHHCwLXXRQ5IB3bRsBGKnvLLA0eJU1A9uVbSkjSgnZ2lfjOf6q4xg6dhDntk6R2W2bVKnwDC5MNfdAWgtdbHNKbTUjxmNsVeK4KPxTdJgrGdACHO1Mzn+T8uhxvXrbSNkpYeo7nfIB4mMsJnAxBNU3KCbBUmu0VyQTH2nBvyLFJMUzYU+kjKzPd5UbSDdqxCQ0I1mptBd6+1+B+AfTEKJL9ZWYxEeI3O3bKhU2aSZ6ydkQQ+MB5RJMdjOuEkgmb+4H2YdoUcUxfgoEs2DT58aX7hZUV3JZYyGmZQhB08YMEkwvlhEQXwxmWBYVMNGtaRd2+cY87TpxRhZLzxrPcT9b7vV+DC2HxVnnGPXNEu4m1WWJwFeNhVmXRX0WjgPzSfoLE4OaqfOUSYKgEsDFh6nIAXxeSxk14hIwqhYSUHj6hcNUEZNjwSkHQnGlU30MNpSzAT3vfHK2jsc8D/IJk0YbRRnuawfPxsl1W82nVj1aI6Y5EKq/2fbqR8GHPTyPEohHBLbMZYkDVdnl4Bik/HR8o/9ETcV4LQBe6uFCRPhl62q+A1Ag2M8rrcr9AdVH8YB5+6r0ytIkB3r5cMqrR26gGaTVKQ2HOkXcI9g8HzZtfi6xgX0Ga0gFU5E7fIIGuPqf2aOeD0nshNEXyGdlqx1G8lI1xNUzE1gR2FDEvLnR1k4d9sXcWqfWSdyv74GaL6G72EBMmfUu2nZKxUoJTA+Ub19ZKDWBqS6xrddHKT1C/t/nbrffTsQSmsZR0p1WRgr9IyqZ4G5WCm2Fl5ADQmwQs0J6WpOvFsaLFsf0JaNExpOfmHt4cOzvKkMN0JGs5W+yMq4H/JGMV7FTt0/2REh5NPi8dBFTlF8+mQYxm2FGdtovT69REq74HJlwPVpX28N6QgOrnRyfiXQNRjfnm2vBmnXgRbKjyIk5Rcb9iGN7iXw+zlD6b5JO3EKRzIcvrs0sdxlVnIhsmLbTWtCBMAjmR2y4/Wx7n9+1EDx4BuAGDeK6tJ/y6YbVe3FWf+Zd+Uc+v+SY4t2TMm2P+bcwPNlhM6FKjp06nMGgf4Lyyn+fTIdjAyDRMNjjFzCl5GYp6Bw4Jge0eyc/jVdoqSuvYMmiFn3j86vWQjgQm57xJMXLRHIALj7uNHLpAuDdVL9RubK3NDzsaHwibH7a5fe2NYFBpxUVl/Mg2vRZDFt7Amnbn8HXQh4czKhjmTGnGWX93hxyyXggASJ7zJCD5EoSDtvyGAJkx1dP8Tn3uYbgDCj8dFnJi6AONDtPChV7oJbOxYqbVHszTk8tl1m/b+qk4T/v7vU2pPWbyYCx/1yCEyuRY/UhwDkxgHUdtA8h7HGBjYiblDIgmHtXHIuYrUU/84GTfHEqZbJmEl0C3DxZYqrCuuBFkXmH0WxlK+nS+ofHxKMRc0jvlhPM02L5jX6Jq5OTBAFKFiU+2GmUWyqCVGTz6nOtjcVbAi7IhuyPuSEyMBwtqdcxqBdxdlYkJNVNKP8W6LxM6gQq4Rbuc/S8PtVkKlYqvZyzK2rJ+BdKn6xyLdvWIB/eP4zQhyKJO8n8+Fdlr/uZejMCPk/PqzX1OXE+UQ1lQ5eC8+fWkNrT4IeOZpAnOCQHtPjVtN1jRfmF0/jkrHf7CPY8OklBRzxMpt9LMWN07N5+dCqnU1aZNTdyYDdnMuYHjmVfxv4ZI/qUSKUTzCgVzLQmIJYIxxFDzyQkRDx0pfc2OzT5+K5ZmEvrKHoTSSKcnvWrjyO39ByeITzFp5jiLzDGFkL6pJ6qy9j0xHtGsmGNs9yaJCXZQGeHBPFHncmxOBJU+k1S1w6sKlCl9OrfGuhMGDLIL2JBiQR3y4ABAKiUa2HSN1x3/hL+0jEHz1ilGAqtE7vvzM9klMmlXiUq1SoVRXg7BvhQi3jM4D4mh5c6JiEY8o3o8KuYIYHBR2uTpsxmDo9rsPzhgAzBeY4SZP1wXaUMAl7uuY4h4GVFuRZBlSjZ4Aw4WmG6yIbzvd7Zwx4fZPHXDS6SqDqdGtU3G5gsL6yZ2+Ydbe4qLZDzvv8N4+6DkS+bBRBDlLTfgyuKhDMjGfLvX3GEDd5KG7s7eaKuYe0aDcuvSxQd1Iq5i61VmSuYsa3F6zvVaoQ56hI9fcDxv2D62Y6pMnIlRIe3fBcpV7wNKfmiv8JdL2oMhu7yYaw1J4vKRDkx7fvbvRkvAc4o8d82C5tzbjgfzaDzmthNpJ5+Pj6Yyg8zSmNgcO+iCjYy/i4Re/gMVQhAqqfqGrzsIs9OMgvePSjOOc0bHAyGJvy+pOyQHsgMJSIsOzfl7UB1WwC4/hBhJd6N+EAlF3ay9I7IZRajSJtoAb9NvuU0xB5c09VFN3VcyD8u4knYyz9xFXbdV5o5xkIm+OGKaHKCaxEgH7j2jSb5KiaELTZlYDMQrhP+5FA6sJyWbjfnZ0SNjr4wI8nc2EB6B5/wHuwZxzqIdQ2sJSD7GMxy2H6C3nB4ptlXV88/PkHhOZ0oAOKkCyxIOIlZOLyVUHn2Kfya9PCl1dcO1OuNm9jGAaPbFjaQb4IeVScUsWAU1wYmRyEYCDrlrR3lJxV1e/3ItqL2wl/X0Nm1rMgKMy5jC2ZR7fNd2AMEyHmBtTGXhC39RrsGKLdDv3AbH5IKmJaSnwsCn3Vd2kZzhFtVe8yICI/FUsRtbwiX7lR77T0iCOZSnOpyz82okkCxaNLwgyQ2dY6qNSFLKkc9hf/vgbRuSy5cgnHU0OSCSDqyXtUTu9+5sxQdAoFUpzKN0pIuwtuzs21GWbW/wXMlZTcXXo8NJCXcNGkMaAjjyW7Ik52rnfO58o/jY0cW3f+2wWm7eKv7nQ5wOsn7ZaOHHcY7ByKZrjrf3EB3w9KP7PqsBbNHVbm7fYkdx1yWV09ZajX8P0bMGoajm1OMM8yQMg2zSqeDEUCcpr02mG+n+sCl/XyJjy8RA/5SO6Hj8H7PQCMhW/J5i4HM4ecg+XfLjhE1ae5uVpCChtYS1CmeLmkvMmnBNLxMD0bJ7T89Q0I9Bm8W4ONTgCWLnVpxEbKMEgHNUzp2XXwJn3A9hHJ2fJI8l7zNlhnxDk3Sfw2WTw9nwN3ITIBGqUDUIf2zQ4Q0zP3YsFmYBGg5oDcmOhbMZpQfE0ZpX8jkZeZWoqY4BYXNnKSAohYzaN3g2JrFKh4kJVf7U8yDzbbxLSxVSakiFl8/mYLmQrs51FI9AdOgU5XmpKjrT/NIfsH8yTd0DVOvJ3gi+9b2vK/RYn2qMv6v1kn5OSWLm0gUyOZN1jgcJ3KmSD89bZaDAYq8HiKjd8jsfaZ/kbD3E1Bg+TxRIhR9EPJm1FTuRxe1MsA2lPjbzDeqE3nn2MOAVWcFdVKKbHoJDxzHuF5jB1mzBWnnEMmnCHrXvSxcc0K0vNpVHmRr8Zb8xcp0uj4j+vNqNxHXB5OFYTqK3IxClyP0nK7E3WDNgKbMnvLdRR5WS9bi/T1fkqbwwbgKGODtoCV+tl9QAOmTH+NrGGN6RfPrxZMl/zU1+HlOnIvqb4Y/SyIMjM7H3KzP0z+cPnzUXjgdq0gYyCNr4PuH1s6PjfjWuVQeAWPrJ4bwRlPUW3DjWthiUHzbjUVkIkB4wA6ysnO2mWaKxF4V9Z1JZAxUL2TIb/tePdiDfzSeNzNFkmaI3CDPxsqO6bz6/BfOXQI9MELdyyTMlsDJi0XdlNDS8GRIQTg7evzlvLhOzNx4nfYQYG2k0nGU2m9W/Y0Lw3b4E7Zq5Um8D9ZxjcbIMOB12JJFaU/MUGu4Ea1NYzaSdYBrk8ekQI7Q4JcQ9a9v/1ZmZf2KkA2+a7yWJn9EE5I8eFlU7JRnQDAjqs7l4vJA9Pn55LTYMVHWKH+7kqLAw1x4fNa1jM7xH1VUpQcAklZR72Rfr9mqsPR9kNVS4NWc3i5e6VfZYI0/rBpcpT5ex1W+ObE3UW2O4yH2D+gRZ0uu+/HJIRlUrIHnOb6iKS0WZAfO6ZnJ9R4Ip3QEm/KYXrbzdrTZGA9O9OmEzZi/7ZtSkHRaGTedRIJGRxbfpUT8Fk8XN4ZrI9vzwe7zVeH7Gtvn3DoZs8NxmuZSt8vOy59V5Y6Dl8qhPQbhS4GqLzIMJajq7xnQk3cwMLyv2wjYVgvd6VEcXP1Q1w+Fx5sLGVy/qzIBE6Q5+1ZVbMdftuSe9v9f+mQsoUmWpMeQOxIZCkOJ7XbUN8AyuEjrpmvFr5AQ0hjrt2x6ZpJmCrb8LxbNvfXsZ5HcG8PzOz8fLzSXtcBJeGyN1QZ55r5Q1TQezQ8zuB5w4mgvSkuQIGdlUMT5jWKZaQFnOwbn8RCn2DKUdP4u5z9ERjV8LfbnmTnhEs7r+GwSvcLyQ+zbWdaemj+CzB4cJcGOv1q3KsXQYZ8vrAHB0d0s5BL7eiTcv+vzXeVTEiLSXa8c7BMCR9Ipq0cLdmj0tYNb3A+UUAJaM3gMfS0vfl6ewG5HbLbbZbmRh6ZbuqPmx7njeo2zKfDT0lHGmKeYFSfWV1Bz4EwoAXe2CGE1zJ7+ppdT0mSaTqN2WMo/nL0sqOqFjVhFHqnAXt+DZ/eJ9+Tfq1eFJ91KdRY2+MUxuybrc5h7RG8ZdiKQIuImMzx9JS5GWxA8GxjHfUjxXWWKYXcNyG+v08/V+lRRMjbuaVFTXycbSnM5DvJRWBPAm6djgRo7f3A9pKTz+M9+zKsPOlkKx0PrbpzGcx3N7R1s6kEXz9ESv72jHUUcvVh6B4C+Kt01B4qVVBwhOW6gTvWtaSlWfMQ93K8RDBc13u9gyCkXM7sPKgkEjn8kpyn9uP/ljn9ZYzKaIAgkQdYHRREGkj8y1Uy6FLgdcvKjjFhMYwWuk6CQ/Xeo5hCOj/NLAh1IYY+KnvFBSpdX2jTAA6hO2a5G/49+OdgyA51eZWrHgkUA0FGhrvkyVRR3O97Z8tHoLc5x0kNaQsixWF5N4FSmb05MdYX2KEXV2vwVOYjHOGlbFcw5JwE6xo0+QQzKSNVq5bH9z+FWVP6Hse3H4mG4IZdTydAYGqTJ+mpp5X1pAOD1aVSTTtxr14dX0fe6jPndcBq9zAmIfzLaAwqlZr3ZtQLm2TRiUxeXwgZqHI0WPV4cTp9KuNvV3rxL+zxKu1JWqJQ6OU0xsGjDZBRIZ+GSGdnNYVh4FFQgkVmAC2Hrmi5nUy5SDHcnHKp0iMlUThUfxcy+CrAwPjgDcaVLIq4Kk2RBRwWTTq0iDCA8UtO40FJSbjCGb+uURlhCqRPZxsA3E0hg4Ox9nb9Bl5r9pUUnEGDOQQ8iK1ZqlRC8Dmc3SvOt9K2zP8TmvBA6jkjiSkjgBxjkFmFA8AjTUqF5SADoEEtdzl1Ml9u7+nZXTO7GoHuHuyVgmO9gYRq+nAYsYZIBI8tL6lzd0h2ZZIFzGZjKPeVa/RLzKjfJbPGSOhhcvnQgF1zkZhQxk+yCRDlwDDVlvyVjbtVR+VSiDQGWqwMqkY4PjzCSMnRH8E5qTUWAJpYCVtlzWq1SpvEFJf/IGzQMn1zmM/Wu5fhDIHpEaZENR6RvsuIe0cfk33WfvZ6lvt6tDCaVEPFpDuLw3EUZrRMthK5W0nuuAoYmMo2/7y5rDYHtnZiKgRJIGkhzK7dnAWRXRJWJx+vEZ1AFX0Y0IrybbPTAPYq+P7MEsPkvqPTxZ9qkR3XKYYR3QAUw6YDZUGOlXxi/yRCUDDNeLb7S2EnW0qkaa/tqHcTkjt+FgZ5Sba0RjiZnbfzfWGZejFDVvL5chXzbQJi0KRRb0Js6i1gZzw5u63SdxvxSkXLBc53l48jFO55kifAI4Vn2QV6aCNhfBf3Zh7nNlbWfZGjnKNPOYmcdPB1lmvSqrBWyqc4Tg7h2lhG3EBtnN9X5EHVEVW/1aN33/jFTqB2/5Pb+F9frYSGYE5BMQnPycCh7FzGkWpMrE6DtkDbmKLdSmSXwhJt0a9u8VhZSJUiTr4aRixB9NY+JntXpmZ4pJOkyvLj3RgKXy8YHR7WGpbs7Cg2TR+zSKnDnGnW+vr4AhmT6AX7BoWa4xxlX/LMgvpyKoFj4kzLnEz9OQLAGH1gE1SrMOtX/HZSAi5VKbRlLAZ2Zzkh7ySYtz7AYGd3BXm9Mcw1Om+aSv1t/WDk1MRGRtQWFGv9O6xGZ1+9Xs5/wCJOlHCMiv2t+pmwbOBqgX/CFM0qJzRqDtIXgGrvMNoTffH/oYKC7FexkKKnpMrxZvCR3Ck+n9Vn1o46mzD6P3HXTXfDlnFZ9K0lCMpKU+5q3k4iLyqX2NX1aPQfH4/q5BpQjRewi2jIyA5SNB2tQxZnOu0iQHM4+DaZkRr2HvJPUEJf/P9Dwy8OEWyDHzEJOvjE1VKQa6UvLGyvNLSPmdVMKZ2UqdylxHSQGYr3kHdpylz6T6gY13EJmGKMbfvjR0q4CjYSPJdPrLyY8rIf2YXXZT6w5RuQwSJg5cMm7IOcz5e+CcVWwqvKnkSjO5Qm5MmWFSeSo46IkF+BZ7/5tJUbzfTGNBnFKw4NDpa+gSpSbhr6EIOVx/VbiI3w58Y5vUsvxe9UbtmdNErGJR1sIIIrD3dhCHBMNVHtMuk+0h5FojHCr9B3kbgyhPoq6rPw5XIh4Fgn8z3Bi5dgOQgfOKuFYHL1yyevqAXzm2PlKbbLTeDRAfS4q+ZOzP/zBsmfrejFCG7DsjIGAqj/sOpnmD6s/ue8vwUuzXHRsc2EGS3wAALuxyjSDIGaaJXpYPYeO8BjlzIkjptfbS1DqwlnnD8HEmBJr8AJPd4TuBvQnX+tBdW3PXlufRP7vBuWB5SPExu3/TZ+KN87OxT7YYRQJMbxRH4tt3uuIFg2fsqkh8sLVGQYdHOsQSkShqGe3XjmfqNJKEEeJrJ6KdNkSpoaaVnq8V7bNdhajjJsRzjOsEbuQd9zi/gvxmDAk9uAzTeXWy31JkICl+06/QAa5ylHlRpqKhtjDRn8nvbgaPk/Yx+aNOgS2rlEAy9OsO4KbIgmYMz2NGBJuDZGrOQBbABc8/4fqYOe/bdJQpKPS/Kl6J+B3BFt02LxLuCHmY3nYMkZQwtZcdYu8FPjHO2xKYyl7z/pC5uQgsApH/v5oJuNq4HBL5jMIwzM+TxDvPhgIvy2sTM5XPxLDYUvEXK+FRZ2Bpo23Nc3kx3I8f5hDM5EjxLIkORtXJeZAQuFZdEpLjNlBqp9mFXZ62m0lo7+dHbHv+T+tn3Ud4Ywi2Mdd/m+IVVxI5tCZrOQr/1hmXdTsyyOhbhNNqmGtu8vfvNfVQ5z3pCXBDeSZXwqxyGFP9+Z+H3KVDbKX8P3LvBaihU/4mHSe7dcd8URXfpG2RsyFm4Pw+2Xuae+YcG4XZ4+yWKdAK6EQQc4vhPVIBe9zY5aW07xgHhHZYN7z4SwyOHkhnPNQtzfuucmPtK3HwnDKlvPBHeakZOYoW4BiXYUvKTOPz7Ne73HpKuUt4fVX5Q4UOMuD6b9ppuKRNXGLmzfGdG7n2WX9DwrewxdPKFvcQgLbhpnxw5Gj2qQmWjDpaP9JGSAbKFTKHJTgh+/8XSjVX24ebRy8sYmthkKBSGKGFUxiOFAxQlvx/R6j2nvuTfIP0lR+u3h/IhGwHLtFrrmUFYoZM0ehEq1e3tOcDP0v4eZfOmmwbW6xsuHhmAlz8kJHtpHW5R3snwAtDWVDqpYRGSTq+lMqkVK99+XF5dkB9MpXgj/ZvONF3VP+rUvqFrXMqx8+jJ2P5wpZwo5v43GjMnG2/y1BAzc4HFXRwMqQcV8r0Mj8w/4nBXR6+LNkdwIjvdlw6pS80vuNUZj70pJI7tD71YgEi7hgm+G0pOrw9bqyZ2QV7XJTxB4FbfljuNVo/DtQXrIkhuyNnoRI7E9gBzhlnvlq5BxJZnvZq0tXOWBgpQi3WIQN6q9x3HqDwsYRSDBTIRX1Xd7m9yGyC0oTIubT3vGKiKNdB0Um5D1vMBMTrd3PgWjeMyQ05JUZy7XvP1g+Rry4b0Fm3Z/kj4ABdmV1X2RbwHl8dzP7AIO2a6nPVP+xA9wYCm6G2K7L7W2tVMwdQmyzdUZuM0SpOa+wp5F9lIzySMo5WHqpg7LoYxEbwTkVCYv5ch1tfJ2Rz3o+MNg7oIUHAg3j7rLbMb89V8W/3YXZr3JFcJ5NrURg4z7OypyWqYYmVmC4QD6Lyp6ThkeFQ8dUNpbK4dwN2N1JKpOOQWYA9ktHmrCWYc1PJ6CBMSbFnS7PXDI8TR1fG4kn5s7EJCm3ZNiOA0vSqr3n1xmXupQcF+h8d8jj9HDyTaW1xw8+CIAYjYmwvCG9yMRtbhDm3Wgi1I/tzE6TRk9bHQQ7jGrsoPJsuMFmymA8jKBwzaK5f4KDc9GQuZUtYjXiDY2xCWVmPo0/Dp2UUAjiuDfuxtX9XoEhZBLrMz2XjPgQMhMVUHd1ICRkmSvKnnK9hGvkaGBN2w/X7YIsMl/C+f+wBcvnO0uD1v2uTvIhCGDSaZG+bkEXk3pDrQ40XkGAWkGrZ8dBLm9QNNXj7NOmiONq5kcvpjlC6LsHmLgAkqq4/+rnTx6C6G54SZZLkIBBJEohvFmFJpdRqYC2cpmKT5hOjnp86CkBNSg/ZgOEfIK/5Vw8DOzT3bA1pwX+T6KhHtzmZ9kDHP/oJUwEibC5OKe8Mle0dCSBMXbrGjp8SQTmffufDWBU5mQ3MpjxUeKjYdFoE5PRUDlIMeqHvRpvB7d+bc5lPVrfFUUFxXu7WVqvKhB8B9gxTlSY2LbqWB2VCFVqfzqi/FsbjA8wjbnbVoHSr5c/9CVSXf/7xzA4uVm/kaBDSB7BfZDZ4VXgBQb3ppyxSMrQOmh5ihWpcNImm2NPQjWXbtHHYLYx1Ec0ChB5c7Rp3dc3vZAlFIRv8W1LB7aVYN9QyT4UOJVAOKTuLGtRaDqL7O3uW6JMv2qDxTgGGiyaNbc01EzUxDaBsZQbeqYk1gJNG8e0Ar4YbiKw5YHjdZ2fZ7IjnRkhUcmrbRTN5F3Ky3wNkGKrOqXEN5bY7eV+vjMEVg2TqInYbDx/KcUxYbmE56NHlvErb8KnqIBwJ5yuXQW1ZvJH7r1S3oW5lUWzQqdfZK3/QW7RvvKWfm+WqYGw+tr0PPybOoL7g06Dl/y4rYqr4tbWb3jPGvvZdkf0MsJ+wgu7HxYFfD4+v3ODMJQXs8w5eIsI9fg1c8QZHhLy6rFwOEzUldVb3cWdGUmO+R5p+Zm1wvCYo0PPE6bhF2ztQMe4EMy6C5Wpl8EC4c5lVp3fqNTo4wbNH6YsMDAm1Q/4hIb5NbDLCPa+WBdVvsFe7J/zXRiGmxMs7exCyDuDwG60UcCFUwRPK63Kv6BzNrrWaL7LjrD7z2PxjIP3tOmNJ5IYmq0G7K3wPwoNbMnAq3TW/PSWbmYOtFgMwpNRLcDmEPP8D6hbK0cs0kkDiX2QzVEufzCuHmlJQAoJQCR9+Y5rmFOk9VfBgb4fZtgJtmB7F7F3aPt7N1yDtvgOkYW57NVIFzYw3C/mzTZ1KI3Qx4WrWvs0QEAl0Izby99lYt0wDGTOvTYo69cmLpb99J6o1/Q6dQ7BBAkG3QFVeFqgVHrPjn4mFwNpGu5S+7MzRRGLiNwT/7ZJPB4Uur5bQsK11FwVxYpSC33x+2H+TlMe7IhGQ1ICPx97mXRqMtvsNPUL7Rp/8lVL7U2HjZyNvx9P/zHT6NHYZbXZRQJFpRr/ue7b/Ad7WVc5cw2aTbLdTzD81A/zrmnXB6w3DPjXMhF6xkXzmPPjx9D3W7M7Fh4vjDN6+bXdsQtJYjzGfKxy+12Fq7tkeNJ9KkSUJ2UPRwXu4pCTwZa7Ak69aDRmc1JXbyQ/hNz3qPaJVqqG98GMy3Oqgs5YeQIdA22YcBQjkTQjxqXnIc+l/lMPJOXyAaW8FqY6u0HdRR0R3pmyMy8tYQZ6cx5UoW/Aol0dVVbV7yVMxjmEAKTj41e+hXuUcHZwAR5Km4fbnX1jHCMIjMK4LjBdN1I6qwY0kNEZ8OLvP+qDqdmxfWz8nuRNEbH33H9l0nvbIq9wi3gQvJnZ6lR0mAbqC7bjFRVfnl/ICzkFwrbcLKKlYw0jxh7N42x+/CcY4VlYLw39yOKLHPJsOeIkmOKG8M3WZqKH2P4e5gq2J8Tj84yiUmpBvlT/5CrOF6ZtOPz2eEa+I9HtUZdmY16xSsCu/oN4Q3wPOvQr5wP/qeIWTXnu4OCsV82qpkkLsBefo6jtwcXzQrE82N5J1rQgFFIdmd2bJjGJC0F3TP47Y6yCvTHd3Fjwrd3v3fqz9FuhYhCT/fWu/aGsHY2AgxHxvcUSdpPxcgp69MhtbmATYj4eJ3ypDRq0QgaMzPXOi6Lh6HK9Vs3pTG9iybzkk6T/nyPhMdT66gwg+XT6S4XaDVyymBMsUkpfwrwPfRcUtMe5mKEKs7FLl/sB2yeXXbQw3ZSqfVLkI6OPlPBnTCTOnCoqN8bj3lm0VkbBbLwvdYN+4tLmZmMVpI7W8zaErM/8OB5dnc7BTcw6vcZExWGlyxBUcgpGQMWvUKsLnDiTRRfXV0aUbydcuN9B2/x6kNWsaj9dwGSbEypNqvtUV2Kt8jzV0GgUjVeHIGbRwo5hLUnrw0Y6w4j7HYvHZVZo7HbL9Hs5OPPcPyXOu2qo4T9AWpx0b7KxpJnqaQN8IkMk7mJxNEMq5g3RCR2uNk4Pt1Pw5hw2jqnW4F7E6QzRU1gJzpbT6/HRAYWiQ7qSn5ka/WzCFBTQ8HHvv1tn6MR2s0E7UzXO7AALtYPuuNz08dvu5N4/+2Y13X/SrK3IvhxvA5cD6ypONkUzubqnyaP4zX/SQuK2CVKFR3XZJ2mnOLj8dMyLmRCMRAe+FG1JNtkDpX9ixzuFh/4FVsvV6Y8AVjVw5MDYwXCejIbjUxTTr/eabh7YLwTw4j7ix4HSq0tqhhjwWUFyyOb4KTJW+J3ynIwr1ur8HDLkhAfEXo1ZPKxhhw4sSDJ8kN1ozV7F7bE0vVgqtrypF3zx9aU8jl6GbC3ajiBB7D5rBF7KJrzzzmNL0x9vdCpbhw0e1LEVdGqpHO0SYp6oX33dJGw+W/QkNR1BGHs5BCy9ZvKJVVkRDNPWQMGwkNC1Tq3YhF9TohfN8sQK+xx0ymyN84uVXR2mI4PSE8Ggzn2i61uIkFkb5hJBpsTL0YZG4KUJGbHoIqlqbntR3DxcAKAT5x0zhR3zCUcTCbHuh8nT6Z2VDtEaxdwUS88pWsS4JGbB6vbTmhoOc05hG5yQPX6G3lMzePV9O0+x1OXSF1LgQVhq2L3IgEHxBXIVn7hWGYzZTROc/VdGoWs8cnDz2nAM8rE5dZXCEXwk3YeEoDpYfIaa8S7xQfVvKh2AT0ebW/1WTc7yc3W8YPNTPODbW7NcBfufZ9g7Kdz11RqfRNXwy6H+gDHLVDC3dAa5Oq8bp+lkHj3txfnAbzDkytRWdiW3hmpX0NwGn6nAZ9k0+SC1cJuJi2RHm6mp9Tkwj+a6aPzpeg4N4hnz/PaVC+2c9MtVUsELYpKq3z9rW8rvBJFjrO/N8zgCQNcgnnt1j/xNaw7DlvkyAYBRl7cQrjkdcwUEuSdxpNt27NYwNrYL9rTfpqeM0QJRTNGYyA1bX2LCs0YJt2oO4sGaPI1j5p8UQ1NVi7xPvl9w0Aj0aJ7bDWlgnCNOzfMDVKjUKhPmsFAmE6ELm5JLtbbPtQJCWr9rlnD7AMcWyQtCHlqNWd47wu2Kv+Ja2DRh5xZO6Ja8xP19h4jBmQTk90gv3Ic7UTHS0fgGWuROaKJZ3CgUQzKC8fLuzGGx3BaNKXDVuSJGL5qGhZAYlEjHWibZGXSZD3kZVqtv7w1CLFaPtxFZHmmUhCDcTVqQ7RNh0eQdIM5VbySMF5sm1YU1zZKGTR3evXQb7gNTOvdqvUhA0vOjVhUrp37e4/WA1x5x4SW5XxQN3jiPV8lLIU0iE4aySmnEnfM2eaXu6TR5gfN8gEB1TX0C2sjefUhrDgnyB7VYEt2d88fpydTD4DyGATaFlsAEjEjk7SpdWu/jMzLO1cWRQaq2Y7KnkAbX1TWvGn8eLn+pebt7zTGTsfdmhaWsuH9TL1yVY7ZBW5CINaeRcMyqsmTAobdEGx7RHvwAsv6KEjAqLOW7VkPamyCTYZDs6xqKMtfcFsLzQhKVKbKBvUR5cWNF4vcX7QCs3HKzhVYQnE7fZmJ2KXExIupwN9CxFy7qQ1vMX+iVwHJ0lPpnBYWjWiP4xw8oZ3xIdYQ4uHQDQ6uD3xqJAO9Mk0RfNCsLdFLTIoaOAi0CHW9lULQWl7MsaN2e5XzQZkG3u5SXtAqDKh8QB2Rp8JmgMdE9fl7RS4Yd+S+Bk4MYbxDsdA4lsdrpxJ9/UeqaH9SrTL19LCP57/TxeIGfsaZ0+4ZGfGPKzlrOa7Mu5EWk1sGOOjyRFyKnYaFDeMk53l6FUG5I3S/ljX1QNwmo7ib64ZwaNZgQIYSA5bkgNrL63Z/9FvbGiU1XGp0/n+YiwTj3Trx+Hcp0+VVNM6cYeWTjnC8Oo4mvaCq7avFCHHOJITk6nkoPW14kOPXUx0MsnW+Yw11O+b20fM9kq+FlmuqVjNPgBAcMEhlYtBiUl1d0DjMIfiyp/7f4T4Lci2hvsWfv8BTVZZCu5xLzaR5rF/Eqxhj/G9T6uxXuC9pvCJXDVWV1Zq8OP9dEVz3TB0lnJNlnxSGFg4xPB8WSnwPpKAYoOQujB67ZUsr8KGiH6YUKqbrlbsVn1UAw7Eau2Wfi1rsi5TPu26KaTfR5j+GEKyzjMj9PRvW4OyG1pIMG3TR0nNMVcujdRXV4sC1K/IoLwYOKDOXWLZUItrq3T9Im5BWaZVeK2vntmMGWvG1Wbq4VUEHqXHWLIAhzDJIWbB3oTuqCKHZ5+m6xbeVY2mVxXzXpMqB1/dAtjq8ItMnjrORSVs5hrhdxm2fPyJ6ngiPqMc0EhW1FrJVWAIYU5e7z6kzhUvadu6Tbm9i4EMgMLrqt/zaVUcyPKMJ4uTHfxq3bpNLVwzzYa8HI9VnYn71HiU02b0GAWFlxlzZ3ehyvwpAzNa5Xs7FmtiMRaDHGrP6zVwDa1U/4Cc5hmVp7LYlnwtp9/lCdLQ1E0XvFrW6edEjSHWvF1DuJ1MJMbiiIf97SArGOz48Eb8ZG1DtujS1YN/wBylF+nWJVTrB5sBc/VN0JN5TYvJJqrb5TLqYr3UOv62/k3d7m7WCb+p/mGLrhwMWW+zHO0mjHMnN73GKh6Ol81AI3Ah5M8EMJmcFoClwFH5M3VdbUWYsW4zNk7ZL/ACc8/4hV9KZNSVHYMZS4H9l6b+O3Iq3e1EW7QhqR4Lt5ruLdT0VpM/6EoFlUYY1X4Ar+iRubl2F9OE4j9s21V6LQM4TCzRZlk0vJwsVYHmqm04fWjgPXKUQR/g4CFdCkU9AbkA/tm36c2URaUDx9Dp2eZYtudBQqSlHTxXiqRyPV40TdrOAEvMLWvpr/b4NzFSjHvtW/M3C0ub6WScRqcmvaNG/SM8aYjL3uUKVNSEgiLBRRm2zCXkg/Nkd7ELw7GtxEPvWehtfBAHkAeBoLijsuB8WTJ52U+HKN0eqKJ7yUuCemS3wsR+/Rki1Y5O+mcx/sQ3yUWM8LnZV+yWaJ8enB3pdVyp0O6Xnv8WBMflZvH3r8RtLLx49anVoEcnJczzmM5+/GCVUXDJQxxGR6JzDSnv1h1ya1yYBDFXtvKbQuPqeFzzwxVFvxzFkRT4BmIIH4Ryf7jFGKUqhqX4P0p/QkUFnn2W9YAL8tCGOW3JrfbBL5dLqUQ1aRz1QhpqIsMWLl0AeHsgqEdjIWsfLRgoBpUadM/XanCL/VPnUBJVH6MU58fzMtSmo8qjLhU9wBftIaR0gk9UQLGqHN3i/xWtZ6xHXUDAMnrQ6SGOK1ZdH1N3D0unBZu3lIcdc0kXR2Uq0xuNhyF4y2Bp7DXkvU4EEWk6mS0D3GBGYYOB1zpMFA3FuDWMUuvLT1qBnhOdsPX+GpU03Gq7Yz3h8Hc+m4fz85vtv2d+GhxsQf4hH1CvHYT2zCuRrOoE6uOhWS0aunWT7wW/36+k2eT7XSMJprxB9bn5R3cW57hUGlFN3hZp5z8j9n8VZTsuV+FNbc3dfLP4NYhKyjWe9X8OZkG2zNr+XyXhN8gPtX8mYuHUoaKDNKBbgU1j+dKtc2sOEB/ZfJj4zcB+HMN7RSjvrCKmm/6j3Nrg8GtqzRRdvJGeO53W0zRDWPhMX0lxlRiyJn5nY1YlqMrQcPTMuCkF5pvbCS0Wp0d+qtAmbzJ+m2nnAC3AWdqwn7PNyoyJ3VsR1TW6hV5Wfzw0dJKUXBCn51KvS1MUFnm9WL+z6pGq2aN3G7FuKEoxdOk/GoHcC8zYZJbL1q+sqU8yfNgnuu6sUNcPStfxx7udHgvSrOh6xZzPn2pLwe/oOpKQQbIePKAqyVW8iILlRGHXPeDWjoBLjRDj6vATzPy2xtJRGnVDH8t6hwQqL8GzeIMeb8xUvyQkOqsLL5RzL1fV9D8ba9+3UivcZ5hCOwdmRvmX7cZzc95fhI8z9EMW1tURdeeTAKBfKj9Aydp8HHBeRfEdIL0uKoWtTk0UcTKkbTeGV+qfL+U0vZzE5UyDOCXI0BS5TBY3Rhs+OSQtL0lvY72Ig7sfyjxSLJ2Kl5y8MDbHqwFDDUUu1cDIYLUJJZaLWIdxdOyxLhBQFjsEQ7X9N0c8RRfeHr3ElkZ3lSdTcRcYXxvHW+jmvUmJbrLzFGIDjhq9NSzmGOC5yHGZEEaBdR9ltk1ukF8sM3QWw5O/H3ifpJ6HH4FqmvMDWzKEoYUF80mbGMWHjJvbZsa2ZyVEQAWbaDVNTQ3YnCRCjRljUMtHOMJ/SBKT6ohTnj/gVZUAUaSkm8VjRa2g2e5Z3QeSsQRLoLF4JrBKqLBaA05kAE8hqodBcmrLWvhodG3AuFuLodHV4H4p5c+HnAZDbdu6tR9RoyN4v04CVT+cVDZZGlRg6Lwu6C5Q6jMX0Jd9mXI1ucsY/ArDyVbuKcDE+SQR2g41bVObxHnhvWK3LfDyDQh0Fph7yAP+sjD2pPYcQ5+7Zl09pX5wSuXukl1E9dVDJu04cMTdZvjafs4GAMGmcCxcOhn87L0ZG4ZNRpOoUTVxD0YP3jRlR8xeCp9L3b1OyIQ2O2HlK9lxIXrG6Xnu4i1LY3VJW7AfRIcIoUb3uj9zU6s8oFSvMOudwKnDxHHZzp8/Ln3dqaC5bchzj4kAFDKi8qhMiI1p0pzD9lYYbR/h5TXZwO3Fp4xNJfHU+JNtiQSU8ZNb5zJc0Dld6oEhxs1MEPNbjLqNE9PhaQ55KDvJUYeNE1PNPJa1y8T2DPxyM0j8ALdOnrxXJ0ojNN28P5HH+p1I4U/Bcw+r/o4jJqJm2/oNWKzaEdm6fxGDAqcBGqAoJUX+t2ALVgNbOqzXTnXg4EKRxShmY+LCb7LsDkEmqYlgOm3xEcAs31P1BAvHq6kRB+sXbn/vO3dpLlHnaVR0ka9cxv+zuq8HDtjKFrGqtg1UVUWKcFBFK8/m2nStgpLGgmWnN6WKqcGP5Ag4kNjvWbEpO+Yri5EBTCRGKqqFsh7hbuUsVbRz98Bgu4cCH3MG3rGq0aNhApyZ3zqA5BCHgbzQbwe6xq4w3TkQqWGX4O1N2+sYrqwpzoP4e5CWgNHaP8VtpkzUPtA0kyc9FgYqa7OJi+4aJB0nn04R5sQNWkuJLUkZpXFO9uFG6rRaY3k6Ejb2NKD2uHQ5pbgh0nZ0qeXk1k6hrpm9CDi9SB35LuYeFLjz3tRu6F7bDgaTGf0BD6ptO2qaacNGjyP9tyUgvPaLSUi5fBPnPNgmRYC2MNmwxzaUWZY+bfiC4I7Hx2ZpjKM9fAFEKq33m3nytVVTmqX4OpUjmQzSvMHH+SNZpCO8VILtnC6ldRZPa7aHEt/ftDJu/rSpxV6hVjVHrjcXgILfBhFqOnkO2x4nTcAo/4bzUm0IMiqyM9ttJAhkYfQN3+SbkG7lw8LdxlRDB1YmOMVF9HeV9+HOrMrcd3r0VUbhwj0RBv7GOzjAN9mqc5SGFcr9hGX/o9RDNoOzlbX/mxkA1KdtfkNZ0RBwPBtx8ep7eJUUnHyQrOf2q44nbyeGHlk4g3C5MbC3k+wYW1dgjYXmx92CHEpPoFCQr6tBuYtaZX1WfdEkZ0mhXGgBC3FQSR6TGhH8KUwMGIbTw+O2I59H/8nIcJiTQOqcR0m/o8sbHtnbRqwy3ZxFX1oPf3wU4BytnyLWEPLE58wPeb+vmwjtcyHwqqY8UWaSQALjQM+BSQ6vMxA4pTEs1ZIabO5EX0dfNvLOmfzGBMXRxyUGsyhD1BS6GCDCYrvy9SKNjUl3e5lpnEjK1vkMJpdTsVGxW/MS7l3ZmS7sgfo8ulq+1GDGJe2bADn9Ym7T+RONRZy4HdfY72YM4D9u0GXa5UDKGhaaMSV76B2MUljjQVyXpNsvXN4T/4Xt4RI3UOiVsqPZgL5SgDDNyA5B/i4hgBK1mUFtgB3A7u7hqm6fiQSSyjz5z/yWEJ6tcQOG4m/Yn4pVoPSE4LcvCqcCzXSOJnf0SnGNbTeHqHHhXr9rSiaTZd1ZOJFf7+bchw5WBTbbeVX0VN40TnHhRf8xddf7VAKM9S0jIsJwafZ65PW/QXQMamuKXmifRA6YFgGwn7j5d0aBxs8DnfSAoHEWHhavkxLx6Sx8cRQMfPFA4KSeMtJ8sH0V6rZRLgKjp1sCmPaN4AyGO1b72DTqzOTbiHu0Pf9hhv3fF/V/jNxIkaGDyJ+RKkzjy5Bn/n5FZx6+bgeK6TjPXk23NT/iBs3gCPT+TD8rDzG8HDNVptBEhl0DijNNSE9H9xmYM/m6jUqL+Dl2ZrKQr3EasT9B1TaQJeZLm8J6JdkKS/RBr/pOjY6/vbb58VWxCYM3nWBF8v2hBHsO/9hpn4LaL8u9qK39xHs1Y6M1WuugOTpkxQFhe3FwaZ3lVvB3PNqgQOE/tWvFpSXL+mSjIa2hYcDX+8mwpJMr4kZNqT3enQ6wtHCIAf5S8nNMlqWg+U3drQWTT0/WKJKH5EzWjFHr8NM/kg4nO+IH6vklOIbeZLCS5iE3kemAiEK+LJH6j2e1XmfYVA3O69gazUkH2OXhxm3+/ZBaj1zM8EaWSmHMyQFnsWfVXU+kApvobfOqR1D9QAxL2/e4T6JlVYU6lnPdt9Zlq0Bsu/lYL+5TSWpLzGmyI0CZgArTz+DslXB9RGIi+E2Y4pBiBvAKDUTWbVUAHpcPzmE13fTPT9cjUK0L3DC8lumL6MVkl/5L6HxPVzYnJZl1IJVWh0n+xPiQQqdQgg8jjgNh9A4m4/7ip2o+1iVgAUv7sm7O3+HlahFiwY0Cpl+W3Q04Ri5a8SY4S1IBwRkUDw33b0B2XoiJxzTsl1xt07HJjs3yWqY2JaWJ/K3e6qkIdEW04ffq/XdwSl1Km47dQFk7ihoEFJpuEuiYZ8Dx2XIU6HhdsXuwuWk+eN1NB95WjfFIlfs7cUk+wY/8Yg26SmI0jmcUKqTqfUP0zDYw1L/fPWmxqhfwGR/7Fbdi8Wc0qpgmMsqdfkKoiy/Tk9YOpnrEgOO1SU4y2DGy2z2N/bvrP5azrv+O1Yd1kAw8ieo3Z9P8Yd3sB7HF5ZEUEUy8GqXGzPVnhO0XGrUsKZhNFFrmFk4IW9OFd0LfnEPTHeRAEhpfHMMzbvqW+ZlB2fLtLBydU3sr9kjPn5JMxCnm4PHSLNC9VlZb8Vfy+Sp3yRRDkJdwgbZL4jLnhKhtUZc6I8lj9Wv7lL8cf6c6o9Nbr7m2iAVUMzLxp+ZEeE5a18jS5j/lJWlAY9vIhhs5xJAGu1BJB9qcuIGdcI7S+pBo0A6IhD4OGaSzDzfZSFANR8DnNDo0JbfVqY5Nk+OEm9YK1vnGpX9KQMNC/nbZSLvTTZ5i3sL8vcGRvaFb0eKZYrqLGfklR98ktJ+zPBAwYjGIHGf2tWJo5RjyZKXTAvnrhANbxf4PAmOG3bclXrdDKkvbD4RhjZ/ZDwM2PSbRlrjXTBLqljb1eQsD06/TbEe5trt8mYXgPaGwNSBdlcpF2jvqn6/7nXrIdjqo/QI0HYcFmgjfEVKZhUv1pG1mD+Pn86F9mGkuY/qZJKMQ7mkbzFXR3p1jB4bUGbA8UqLz7AOGWMFMBVnZ69izqmQZKzyR5C4GEHpnWR/UWD0iSJ9vi1ANRj03ocpDJ0LdXmAEh+8Sb6HVI7b0WPtfhNVqSv7cNHbDYWGcGMhHB5DtAfVNmZfbmnqIIKPtoWyq8M3Sp5mb37hjQL/ui6pP4L/L3tYBFxcXPHXVNtWWhP8rfBsHazW+Qv0uh6spPQHQiS82V1n+IafzwjtGgGYLE6VUfbYGGz8wtGfHTUbCMzN8qHl8PoRA8i9nS2tqPtTy1kvI/RpZL9MnvcMZ5xnAOYqhzTYdlK8jgXpu81zcXAL0Mqk59VAOJMEHzLuSmxNRoGl40pCDBbSjLZEKk9HzhV84JEkY2aEp0cI55v6iXvc8ZGj8w6ujUjdv3p5CiMyiNcCr1UkGnrgyzgdtHVrk9Iblm0CuP3zIqVzEhM45h3iPhH0J7Ol0IFxkxm/eP4ZR3cnUmtxCpUZpBOOyxI2B6n+Qp3m3P0hXlnIkXp5Ga4rgMcewzRkh5KPXYyXNd0TWWVD7+ykD1CN8QC6Z/dH4dI79OU2c0uvXy/jYYNijl2J1CsbG5Mck1s9sSoTyQvZkEE6Ng2KHLA4Nd37VA7VXPXzUpuuo1UgTDn9okuV+oF71u1TOtlNG+Jp0x7Ztu+SmZtVMBLXs+eM8JbIeNmYM1WhRyqfNw9v5wMPJUc7h+rN0+wc0K2nRv2DEqvHVbDwG/QIP9EpTgU35HZbzEY/tvF0Imt/wtKt0Tg/5qgkWFwOUqH1QKEHx6kAzaCfz6ohDOm2PjiDNLDFLIfp/+xMgbIHBoCvbs1AgxFqdQD26KY3QxsdZvHUFC6JfcSrm3TuzTmejLxA5PkcGGy3W9y6HPBoQA2wydZa3R4Hz3wGu/gGNUDk5oFBw8bVjsDkx6oe8AWttYgyaggxXbPQIvyrsM7M3mg4C9PxtGa9bBfpAdXnb3fngo8NGrZ4JiKxj3ns1W5Ak3ZB3K5mCV6fa4CP7sMI7ouvLPhmNSfVHbahYhS+g98ZTTYYGDNVq0Il8UHOcKn9oUtnxeqSu1t/7cAAO/BcG0B4BkUYk7mgaZL0ozaN0kFTG7JEpM0tY9w/8RmbuL5tPyIA==

xUn0EInIAxhIFU5MBKd9H7Gunp4t/j+IMvYa9Ek45i4yMRGZmJgCgW7duGDhwoMQRkT6yqjnq06cPbt++jenTpyM
9PR2NGjXCrl27UKNGDQBAenp6kZ5HNWvWxK5duzB27FgsW7YMgYGB+Pzzz7nLMZGLMzVRsDShsOX7iH1tHetPRDP
0GmKStQ4dLI9FjgYNGoSDBw9CpVJh9erVButlSVqySo4AYOTIkRg5cqTOx2JjY0sca9++Pf78808bR0VEcmJqomB
pQmHL9xH72u3aqVelZWWJfy+FQl3k3a6d/nPslXDK2auvvoqzZ8/i4sWLemtlyTHIalqNiMga2rVTX+z1/eGuUAD
BwYaTAanfR+xru7mpl+uLpXn9xYsNrzizV8Ipd76+vmjTpo3UYZARTI6I6IyOW4uamXugMlkwtTkwEp30ezVF7Xsnx
Dr92jB1CunOHXLr5wKijItGX8tkgEnaEhZVJSErcFkSEmR0Tkknr0UF/0q1UretzUZECq9ym8VH7xYvWx4smVvtc
+cAB48MDw6xcUAIsWAZs3A/HxQHKyaTFaOxF0hpYAf//9N5555ThkAiy2ltNCtxbjci5yalDtr4
mjgqF+lhUFNC9u/7X3rJFnWgYs3kz0LevuNgKx1i8KWVwsDoxMjUR1Pc5NYx9Tkfw5MkThTIaG4vjx4wDUe33OmTN
H4qhciyXXbyZHRjA5IiJLWCv5UqnUIyf6VoRpRpCi8q/lTk/W/dj6tSpAIB
69erhzz//hJeXl8RRuYmRzbE5IiIzGXNbUOskdhoEo+0NP21SsYSLFsz9XMC/03XWXMa1BqOHz+ONm3aID8/H2
ubjh06BCee+45qcNyOZZcv1lzRzERkA9buFm2NpfL2KkS3hJil/o7YXPLRo0fo378/8vPzPzAQAfffQREyMZYnJERGR
ltugWba2l8vYqRDeX2KX+1upmbi0fffQRkpKSAADPPPMv74Y4kjInOITo5efPFF3Lt3r8TxnJwcBbvGr5X
I1myxbYg1l8r36AFcvaqeghO7Ks3WjH1OfRyhuWR8fDwLL40mOTXRylJCQgLy8vpVwyJycHAkWZM0N6fPXs2nn8pC
OkroTEUnIFt2irT0l5uamrk3q21f901FWfRn6nIZCIoxYvHBvH3I6dOntf/9119
/ISMjQzc3tfpVLhl19+f+QbX19RZ+Qb
3RkpKCtq3b48xY8ZIGxBZxOTVaqVKldqfrKZ6enliyZAkGDRpk3QQlglxtVqRCSWrVeFmbNUXtfKOQ1HXRKv+Zz
ff69O/jT9nDQccbXa1q1b0bp1a4SEhEgVKlW0j7m7u6Nq1apwzQq1apwc6Q/PayEyRE
RmcNYI8MJE4B58xwjCdSklGcNZpLkmthnyMbYnJEROZ6/31g/nzdj9krGTG1qaIj9Dgyxl7dzE0lCALu3r2
LSpUqSRcE6WX35OjgoYOYxYJEhAZmYmCgoKijIjw2ZcoUS
Zs6fU4VAxlly/TS7I1li9ejVGX19f+Pv7athaBgBX19f+Pv7a+uQAEChUDhdzckRE8uBoowpilvMXTkas/TnELnF3hCXxcnDt2jW
MHj0aOTk56NWrF44eSYsmuDkRNiXiyL79kRsqpkCbd8UIWn4e+Pz3vDGPo9vDRSzDRRL36p1nSBxX1vZoqnlW2E/ew+4DTnEfPPpYIsgXnEpeXhEggCBFvBY1+OqVirKPczsQk0Hz5ce3/kiVMjEgv0QXZQ4cOxbhBhwDlyhWEhoZCDg999MD
AgQORk5MDAIiIGBi5GRE9zm6e/cuevfubYtIhEs/Y2HdYidjm/rT6HpqmiMWKaSLq6JUuWID4D4+HgAHByMzzS
NmchpiE6Oevfujd27d9siFiIiUWyxTYi1l8r36AFcvaqeghO7Ks3WjH1OfRyhuWR8fDwLL40mOTXRylJCQgLy8vpVwyJycHAkWZM0N6fPXs2nn8pC
16tTB5MmTceTIETRu3BiVKlUyenzFihWyJSgvJZuE1dURwVOZRExdv9Bzo1SMDRDK5+sTi/qNbbQJE0yRaB2
AMGbMGG6b5aREJ0erVq3CggUL8Mknn2DZsmW4fPkyatWqhS1btmBYvK8wSGYbaC8hRTK8BU3l8jqvm0DkTx6d8eTQycnG3lHZu9M7AOrTM2Dt
BAPXr18fs2bM2bMljohsRXKkcsbT6BCqJZSoy9qqDr45cwAMXGsSEc7RryDQ4XHNaAOC3GRq4E0djrHWPdVFBW1rvuWmB2ZLLS3Xe2vJIKGJuX0BR0RlyGL
IVkQnRxp5eXlXltITk5G7q9UbQ02S9DRGQ2TV2PsWw06pC4yNpaU2OJzOFprA7m7desWfHx8cwdUUdcnebkRBd
kP3z4EIMHD4aXlxcaNmyosfDsi9aNT8lGGjF0LxPxBSplBqzyMp+Qz8S12+EnYdsbcwgQ1OsN9gvBXfYKaL0PoA76o
hIdKt3HMGXbt2xZkzB27Fh8/PHHHUodNiY6OZo0anJ2IWJRIPbGpCaMQH6VA9sKx2cDcXCtKZ5IH7dGUF5jbfLDgHUHcqt9RQE7Lzs29RZ4V4/PPbbOlkmhmAnIE2w2e6c7WKUmmyGNpIxgIQyhxBo
BQY67V5iu5GXsWGD8eMs/h6O2NnAGgYGBWLhwYmFSOR8RG8fUqNGDWzbtg2tW7dG+fLLlcerUKdSqVQuXLl3CM88MP
8o22K5Sy4fQiRPMhlGknfJrCaEaJt29SF0uZ8DkfZssRZPHnyhImQjNl1+5Bbt26hatWqJY7/888RbYSISKyJzk
UGxvrZ6RQAOPGmZ+8OHJrAzkanNmwYcnNzsXTpUm02uzJE1T6s9++yz2Llzp1aapY73IicjxavXoi2rRpY73IiIicjJkxRy
O3tpATr777jvExsZiy5YteP7Y956GSopMoSUb0yNHs2bPRuXNn/fO4fDhwyX6gs9TbhXhGunOHzfbdMm5vhEZ1OOrNLp+5
snbzIobWBHGRmZmLYsGHa+//z7cOA/pUkSPHIWGhuLgwYUgGzwMCNnD5+PAhex9GixYtbdDhRrU7wFn9IjU5wDKrQx
CzNbJi5gtS6yt8Oq7hARptmyxBkEQMGzYTNy6dQsA0L17d7EEgcFdmb6IJsV8OCbCIyVkRt6moyTcG0sX5GlhR
Ma2IFdG9ZYosVfHFxuruAf/aZY64WNCQ2NhYDw4EAFSpUgVr166FUqmUMEqyBVHJ0caNG40/X7RJARj7PFMc
shRdeEPtyDo3JEZFrs/YKMHskL7qSFVvtn2atxNERXLt2DY0bN8bGpwNQe9U/S7N2bbzwYcbxxX8FtCP8vrCY9W
aNRR/qkKhcLqiNSZHRK4hmK4tIUE9hhVGzfEbzxt6EJCMVvSCOJQHSvkGgywIABA7Bu3Tp4eXlJHZQgwkLw8rv4eH4/fPo
kJFq2bImdO3ciLAdcEFZpM8Vf2yUl1xN9HEjhazsGM5/M6XwJq2E9iYpg4dtp1xAPH5o+ApTrdvo6gKBvh/gRqwm6JiTg5W9pSS0
W8RARORRbFVHLoS+TMc7SOkQBBkXU6O0ZPsZf4yA4L3
XWLNmDZKTkxFmJwcQCahBbhIY+ggRNLzFmMxr4KaWC+LxSGXRTk7Bhr+JL
oFBgyZIjUYZCDWEF2QXapUydIZCDEUkoCjUUdZZICCoUCgiCwIJuwNJY9V4DJiRStA4hMYdfVateuXTP4f4IXS0BJWpRNx9k0N7NpUFNA4OiYHBG
3OSaOjx49wmuvvvYbx48jS5cuDUodNiY6OZo0anJ2IWJRIPbGpCaMQH6VA9sKx2cDcXCtKZ5IH7dGUF5jbfLDgHUHcqt9RQE7Lzs29RZ4V4/PPbbOlkmhmAnIE2w2e6c7WKUmmyGNpIxgIQyhxBo
xePFiJCUlQaFQoEGDBhgzZgxq165tzssREZHMyWn13W+//hd5ex//aZNjDw8PPDqq69KHROFv
p0gSNGjXCH3/8gYYYNG2LPnj22i2iJGIiMgq7t27t27hwEDBmjvz5kzB/fv39fqVMhvNAuIHJIoqfVmjdvjk0ObmHOnDlDLFjn/wwQf
YvXs3/vzzzz6sGKDVOqxGRvchtakqOIiIisGHDBgyAWFgY/ve//+bHWV4+k+tX8jBeUkfRG9oDfu0UrYCJ6g3MP8EBMEY6K0b6cdZnRNc/QUk2oHqAU+PT8/0ry8E3Xv
N8PjxY1EBODomR0RERPrZWw/W8DJZUYXTjREDmMZD4JkAkS4Gh4Kb3oGMm7WM3uGsYMSK5+Z5T
MmuXwxbfiSEtTH4+LkyYuZ3Lz5k0MHz4c0dHRkHZkeMATx5P+cO/yPRWOzk0biSg8Go5s3oAQj3mX7mXIpZGe+FhPz8QeVlNyPyWmxvVP1YFMkM
bxEhE5LRUKvWIka4ZMcAfh+zV/wz5CUfJ5T2a/ZB7HXTQMJzBGQQ0T52IKwP4jZSz9u4YtI7TuBJfOTtaHM8OWwpEmJ9TSzJYYrA/d1/nmgNOFZCB
NAEBgYCAmTJiA0aNHO91GmrRpUVEzVtSCeqcVakvHTpL9WDlGyI5+HsH6F0ePz/lmOEWs5wTk4cdn43pMbyR1CKDk1IwOBhj8d6PuOyS05ZLLIxIKa8
jx+L69evIzs5Gdn3t3fwrh0ztAiq0Aty5rJm5QYgkqWEA3jJ9KugGrdjD5mPRaNCwfhTHEk9xP3/qdpEMvUeskcFTVvrbHX6HIytHNL4vrxc2yRMRGXMsO6c5d42zlO
zVI2VSp3xbdmi/Ml+O3/4d7c3OpyTG5SV9IBhz4aFc/sQzXr8P6djyp6RvHeEOP4j+NMaWbhDf2hGmK1h2ZbjONnmrK8SMEiRU6TWg1O6iuUbhy0DsH3r0BS9jE3vXu
1/PLLLwCAYcOGITw8D/++KPB53Xu3Bnr1q3T3d7dpnEREYjjL5q0AzK8qt8MyvZCQEMTHxzvd7AbZhujkcVRXU1trxRnMxsHb/pqugsnHfuVt7kmx3rhqufcd
nB++99x62bNmCoICvv69bvtbJ4i/l4zBnMjsfO89/QlrDAQlQqUvAAo6P9ngJ0vXEPbt7I/hPN5t4BAi3eEXDzzUqMcDf/XMoHt1d5Z7LKqIH1jHAWz6PxgRn9u8/3o

SCX9/f6vHRERkCU0+kJYG+PoC/5bDlCCbzVs1VeXFKzU0VeX6Nlmz4zI9JkZkKtHTakOGDMEff/yBnTt34t69e8j
OzsZPP/2E48ePY+jQobaIEYcPH4aPj482MQKA1q1bw8fHB4cOHTL43ISEBFStWhV169bF0KFDkZmZafD83Nxc5OT
kFLkREVlTXBwQEqKuNXr3XcOJEaDeo8yhi7GNVZUD6qry4lNsNlym99dff2HEiBG4f/++2a9Brkt0crRz506sXbs
WnTp1gre3N8qXL49OnTph9erV2Llzpy1iREZGhs454qpVyIjI0Pv87p06YJNmzbht99+w4IFC3Ds2DG8+OKLyM3
N1fuc2bNna+uafafHx8EBwcbJXPQEQE6M8HdAkKksmu9gcOGP5AggCkpqrP0zA3oTLBkydPEB4ejhUrVqBp06a4cOG
C6Ncg1yY6OapcubLOqQtPT0+UL1/eLOHQqPQeMuT4ZQYlxJjxc7mV8Uz8EWx9PLqLLnwqzkDfI3dlAzIAeAxdnUaZ3
XLhhM4iZNmqQtNM/OzkZqaqoz0RE8iBF3bChfABQjxRVqQJ89ZV6dVpysgwSI8C8qnJzEqpCDP3+Zs6cqd2twcP
Dg3/kkmiia44+/vhjEdqQGDQj4t0IwIyMDEyZMwOTJk0W91qhRo/D2228bPCckJASSnT5/GzZs3Szx269Yt+Pn
5mfx+AQEBqFGjBpBi5evKj3HKVSCaVSafJrEpH8SNWN2pR84NYtoFo1mS3bN6eq3IJleoZ+f0FBRzFr1iwAQOnSpbF
x40Z4enqa9l5E/xKdHH3xxxe4e4dOkSatSooe0umpKSAqVSiu3bmmTRtkZFyhxXac43ts+br6wtfX1+j79mNG
jeO655wAAf/zxBAf/rZRbkAyOpaByWLBQC/xKdHH3xxRe4dOkSatSooe0umpKSAqVSiu3bmmTRtkZFyhxXac43ts+
fIjCwP1T/DiNNnjwZLVq/fPNNu38GIpKe1N2onWrZfmFubuphm1699J+J+qjP7dHDeEK
lY5me8TKlSbhx4zwA4Nlnn8WkSZMs+1zkugSZuH37tvDOO+8I8I+8I5cuXF8qXLy+88847wt27d4ucA0BYCt26dIAiC8PD
hQ6Fjx45ClSpVhDJlygjVq1cXIiIhJSUFFHvm52dLQAQsrOzrfRJiEgq8fGCoL6MLMGr7Fx9vm/fPzBSEoSBAUCt3
vq1AIQnCw+jxZ+uYbQXBz0//FFv+A27erjxX/QjTHtm8v8vKGf3//EwAIAAR3dw8hKSnJ/p+fHIol12+zm0ACwIM
HD7S9jjRstcVGpUqV8NVXXX9sj0AEmLFaLTk5Ga+++irKli2rXaFWsWJFVKhQQfRqQmvRs+fCHk5qFKXy1BRe9R/mo9zq7GEaR
vv9VdQLR4cYn5Tf2/l3sABmjvvf32XNSvX9+sj0AEmLFaLTk5Ga+++irKli2rXaFWsWJFVKhQQfRqQmvRs+fCSEJKVFZeSKES1
NPlCtWtHjslm2b4g52WePHsDVq+rleZs3G1ymp//3sgKAJrl6ERERo0qe4oDbmpDjEj1y9M477wAA1q5Cz8/P3Y
cJSLZMKPMxSZ69FDXNdl4xwzMzf7dHMzaXme/t/f+wA8AcxBQEAs2rcv9ne/VMsTSbYUgqDr/yL0K1euHE6cOGF
wyw5nYsmuvkTkeDSrnQQDd01qyH72Rkkqlbv1tLPtMTjY7EzT0+xOER9i+3bPo70/f8jb+wp2eJddv0dNqzz77LBs
jEpFsOfW0ltQ0RVXAf8mHhpWKgz9/kokRjbswk3OTfTI0eXLGYZYl3333RoO1Qpj0RjD0T0rRwBGK8KpGUhOAtb6r
5JztsBO+6dE1jBQfrLLI217Zt3yAjww9Vq76gg/eXkKDewM6Y+HiZdd0kU1hy/RZdc3TkZQS/RZdc3Tr1i1cvnwZAwcNBT
arTxUzMCJSAZMLHMhc9i4qCo5ORlDhgzCP//8g+joaMybNw+lSUmYCJF6eSLJlujkaNCgWjevDm2bNnNCgmwiItL
NRtlnQUEBBgwYgAcPHgAA7t69i0/3OSAYAJ2RmxGkuYuPh+ODCT6sFgGSALJhjDdiuzLQ9JJKFAnMeMqIzOKzFdnQTOIt
tWgRAz3fjKMsTSXZEF2S/+OKLOHXqOqlC1iISIsIKPeLi1avBwKAAfv3UP0NC1MddxbkS/Dhhx7eFJTwHyz:jJdzrF+/Ht7e3vq
/m+9tXyBOzkn0yFG3bt0wduxYnDlzo0bNy5RkP36669bLTgiIldgbERIys1yHUVeXh7Cw8ORm5sLABg7dizat29
vwnfTAz1EdOEmAsxYraZ3bhdwyoJsrlYjIlsy1p9Q0zqo8OOFWaF1kCxMmTIFM2bMAAA8/fTTTOHiBMqU8TD9uwH
nJLVcZH7Wkuu36OTI1TA5IiIyTjoUYaVKXI3+xx9/oG3HDpmCFi1ackW+OVyoW7hdm0ASEZHlTO1
PmJZm2uvZYzWfNOz0aqrYZh6kGlhhhHhTigfzoI+G8D+1u3THs9W69GlqrqU0U3eMNBc3XWrPp/D
666/jwIDmlHTC0IlwiCmbJyyhHMxhRCO4BIkOxxtXn7WNWLONSVOI2m8nuGdqs/k8G1J6RIuFA3JHFyLBDVenoiObpR1
XePnl1+H15YXRo0dj1RIgaJitmdjLExmWTHTUUsxE00NoEmgbNPdb7wnLw24+kKL9JPDLmUQ2EsBHiPJi3BtLs4T/HilGqoVm
1aVejO8qAQ+nSpeHn51fkmB22cnnMeJIRGxbZREJ+E2I3wTNdmdgsXq7hVHiufEV7K7MT/Xt2fVR
KNeCwadMmPHr0YFDs3bu3IfsvXv3Ijg42GqBeHbHHZRZEzZmVrh7VXD7IhRwE/3DRDUVO8tmMmuvhE
pE6HCsega9bDxdmV6STHgsH37drz77ruoo0aNHr3N2sjZWsdAXdRJ3lbBGhF1aEKtDEmjzEzXHXJkBnfSE0GK5Aaq2TWP2tWtR7zKDF3uP3DB6Ku39z5FwYnGQzP
///47YmNj8ZlmfJ0IyEEZmiYqztZNFjWjHqb2J5Ris1x7DzhkZGRg+PDhAIC///4bSUlJRpMjgBsJGyU2G3dxVU
52rFjBxYsWICkpCQAkCkCQAQIMGDTBhwgR0797d6gFKjX2xNOzRZNGe/JpmlAaG3CwxvcjCAJef/11/PTTwC
Anj174ptvvuEG59akq89RcLBTdgtnE0gbYnJEZ2P5tmbJbfqNYFmjFWuFe6sJ0KvDktZRQGbYhBAPj5+eHMMmTo
oUqWK5S9MRTlyNm5Flly/RU+rHTt2DAUKbuRhgTnYtB7pmBsdp6d3wJNuhqqhlxFXIWaqpycsgEfasGNnbZUpcQUFn4c9pCzxzXvOv+IMNYamqBWaRaHy8M+V2kewUpx
EyJxmkSqVCgMGDDBw+DBw8AaIMhd9A5a5KprIXkRPJC5ct3ynLG60FcvMqAPeHI5MrvW93rTHgqwSKSkdLmYmYBxniPrR
YWlPjQtZmbRXnK7PvGDBGowfPx4xNP4AeBISg1OnTvlH/b8lidt1bTalU4kWDqyyWOp6enF+lcSkTkqPT1xinO0Ku81uo
sLYeRJ3ObRV64cEG7JHYhcocD69euZeZGJHkcRrNmTs7GztsXv37uHDDz/Ek6+8Gi+Ihspfg00bR1m0
kaK3O0lLtaSaWuc0ia9WqhalTp6J6dKdKIjo7GCy+Yn8eNtAm71xcmfAm7WhfdEq3o3mhzZsDABITE+Hn/Y
4XSNITqsRuQ5rLuIxtWWAoVVw5k5TScHUVYCbNwN9+5YXtXHkV89aDYpQzv+ZCfvr0adStErwTUfH2Cek9KXt+kKYj
UKXh6emLgwIHo27cvypBQAHRg1ZkdOVWjFmI/zd7UCL5sJk84ZsYtsKqoimQxjY3X7QNxfmz1rp5uebfTkBQDMDbN
2LERETsPSZEHMNJUjrMoWszvFw4cPkZSeUhBYtWIWt2mt+zUg/UCITiK45Ii48rs0s0WBt2Z3CqDkZy5e2D5x4kS
0atUKFMwZMnTyx/cyIrY3JERGQPDYpIFXWy1p5kt7C7z1rQISXNi+Z88eLF+vFA5SPVNI+NnXkrTWsCeMbWv/Sbox2ZNgOQ4Z0XV5ZTbI/6CAi8bvbxnmFAXwWD5J
FuH2IESzIJiJdTC3eNncrK1vsaWavAm99383du3fRuHjNkR1xr8XNIsZqwWpSdfUpKIb5BH4OBCLfHs0M5hMNXGAVh/NXQV
nzT3NNMmWvjome2ywGx4ejq+++gNAui3ML7/8gd3u3Sxy/9eia7ZREtOxpz8GjpsPkY+cFmEj+Jxue2d9gZPpc
VcH37qn+amnyYMk1lKnP7EFnLt99+q02MfHHx8Wn1/ELWr/o89F1tZkw3EBxCN8ZTQ3bw+IcfEk8OT3iSIW+I6HS8cvPzc3VDpUSETkajpu
pZowtC7yNycjIQGpQGRkpPb+0qVbvVLvEl+RqUaT9yP5h4QmxuT+/7p4ks5K+GQ7J9tQ3IRTq/tHTNkMn6Ah7z4uAEzOPdpcfKtbbbTgrNkH4gWv9v5Ykk2Ikk
i1fJ6a/RfslWBtzGCIGDIkCHaHaYeGXvPXvinXfez6b2IOU2SXZncnJ0RtvvAFAvfdNvrtmaF8fdNRERExkcfKlCmDkJAQLFiwwwKr

BERE5EjlfH8X0IbKmNWvWYOfOnQAAPz8/rFixAgp9/Q0cmVTZJUnC5Gm1goICFBQUoHr16sjMzNTeLygoQG5uLs6
fP4/XXnvNlrESEUnKqtdHOy8Ht7S1gLmeffZZNG7cGADw5ZdfwtfX17pvYC+WNq4iWRFdc5ScnFziH/e9e/esFQ8
RkcOy2vVRot1krVngbaqmTZvi2LFj+Pbbb+X9B7RU2SVJQnRyNHfuXGzbtk17v3fv3qhUqRKqVauGU6dOWTU4IiJ
HYpXroznL3azIWgXeYiiVSvTs2dN2b2AvUmSXJAnRfY5q1aqFr776CqGhodizZw/eeustbNu2DV9//TVSUlKwe/d
uW8UqCfY5IqLizG3s6BDNhuwgLS0NVapUgbu7u9Sh2Ia5javIruzaBNLT0xMXLlxAcHAwxowZg8ePH2PlypW4cOE
CWrVqhbt374oKwNExOSIiXcy6PiYkqKfQjImPd4zdZM2Ql5eHVq1aQaFQYOPGjWjYsKHUIZGLsuT6bfJqNY2KFSs
iNTUVwcHB+OWXXzBz5kwA6uWauvofERE5I7OW18t5uZuJpk+fjsTERABAREQEjh07Js/VaeTSRCdHPXr0QL9+/fD
UU0/h9u3b6NKlCwAgMTERderUsXqARERORw8mXgx85cgSzZ88GAJQuXRqrV692ncSIU21ORXRytGjRItSsWRMpKSm
YN28eypUrBwBIT0/HyJEjrR4gEZHTMLPZkByuu//88w/69++PgoICAEBMTAyaN28ucVR2ws1onY6omqMnT55g2LB
hmDx5ssvso8aaIyKyKpG7ycrlujtq1CgsW7YMANCqVSv8/vvvKF1a9N/f8sPNaB2W3TaeLVOmDHbs2CHqDYiIqBA
Ry8ElXvVvsj179mgTI09PT2zYsME1EiNuRuu0RPc5evPNN/Hdd9/ZIBTDZs2ahdDQUHh5eaFChQomPUcQBMTExCA
wMBCenp7o0KEDzp07Z9tAiYiMMaHZkFyuu3fv3sXAgQO19+fPn4+6detKGJEdcTNapyU6ta9Tpw5mzJiBQ4cOoUW
LFihbtmyRx0ePHm214ArLy8tD79690aZNG6xZs8ak58ybNw8LFy5EbGws6tati5kzZ+KVV17B+fPnUb58eZvESUR
kEiPL3aTa5FasUaNGIS0tDQDw8ssvY8SIEdIFY28usPrQVYlOjr788ktUqFABJ06cwIkTJ4o8plAobJYcTZs2DQA
QGxtr0vmCIGDx4sX46KOP0OPfv8bWr18PPz8/bN68GcOHD7dJnERE1iCH625BQQFq1KiBUqVKwdvbG+vWrUOpUqI
nJOTLyVcfujLRyVFycrIt4rC65ORkZGRkoGPHjtpjSqUS7du3x6FDh/QmR7m5ucjNzdXez8nJsXmsRETFyeG6W6p
UKXzyySd47bXXkJWVhaCgIOmCkYKZqw/J8Tltip+RkQEA8PPzK3Lcz89P+5gus2fPho+Pj/YWHBxs0ziJiHSR0yb
woaGheP3116UOw/64Ga3TMmnkKDo6GjNmzEDZsmURHR1t8NyFCxea/OYxMTHa6TJ9jh07hpYtW5r8msUVb0AmCIL
BpmSTJk0q8hlzcnKYIBGR3Wmuu716qa+zulb9S3XdvX//Pus2NTSrD3X1WzC62R45KpOSo5MnT+LJkyfa/9ZHbCf
UUaNG4e233zZ4TkhIiKjX1PD39wegHkEKKDTunJmZWWI0qTClUgmlUmnWexKROHJobiglR7zuXrlyBc8++ywmTpy
IcePGwY2/MPUvont3/mN2IiYlR/Hx8bhy5Qp8fHwQHx9vtTf39fWFr6+v1V6vsJo1a8Lf3x979xzuzRdmnNy8vDvn3
7MHfuXJu8JxGZTi7NDaXmSNddlUqFiIgI3LlzBxMnToQgCJg4caL9A3FEZm22R47K5Jqjp556Crdu3dLe79OnD27
evGmToKgX

YOXIkUlNTpQ7Fcehbtm7u8nh9VCr1iJGuREtzLCrqv+X0tqLpiaSvgNwplyhaH5MjIiID9LWKcUQxMTH44osv0Lh
xY/z4449Sh+P4rNkxWkytjy1ZO+lzUUyOiIj0kFOrmIMHD2LevHkAgIcPH6JGjRoSRyQTpi6PN8aRan24TYjFWHN
ERKSDI5SPmOrBgweIiIhAQUEBAGDGjBlo0qSJxFHJiDWq1R2t1ofbhFiEfY6MYJ8jItcjt1YxI0aMwIoVKwAAoaG
h2L9/v+sWYUvFlN5KjvSPxgVYcv3mtBoRUTGOUj5iil9++UWbGHl5eWH9+vVMjCxhbpEZa32cCpMjIqJiHKl8xJA
7d+5g0KBB2vsLFixAnTp1JIxI5iwtMrOk1kdOlf8ugDVHROT0xDYsdrTyEX3ee+89pP+boXXq1AnDhw+XNiA5s1a
RmTm1PlI3jqQSWHNkBGuOiOTNnOuOHMpHjh49ilatWgEAKlasiLNnzyIwMFCaYOTAUIYsZZGZvqRMMxXnSJX/MsO
aIyIiHczdnFwO5SPPPfccfvjhB1StWhVffPEFEyNDjE2XSVVk5iiNI6kEJkdE5JQsve7IoVVMt27dcPHiRfTp00f
qUByXKRmyqcVj339v3dhMTcqWLGGCZGdMjojIKVljMMBa/QFtidP9BpiaIVetatrrLV7832iTNQqoTU3Kxo513O6
jTooF2UTklKy14syRdrO/dOkSDh48iP79+0Ohb+8s+o+pGTKgHhI0dC6gnlONigIKCtQJi6UF1GIq+h2x+6gT48g
RETkluaw4M5VKpUL//v0xYMAA9OjRA3fu3JE6JMdnaoacmflfkZkhmmSqd2/xhWy6GNsktvh7A6xBshMmR0TklJx
tc/L58+fj8OHDAIAzZ87A3d1d4ohkQEyG3KOHOvEwlznJi6HKf33v4SjdR50ckyMickpyWHFmqlOnTmHKlCkAgFK
lSmH9+vUoV66cxFHJgNgMuXt3y97PnORFX+W/IVJ3H3UBskmOZs2ahdDQUHh5eaFChQomPWfAgAFQKBRFbq1bt7Z
toETkMOSw4syY3NxchIeH48mTJwCA999/H23btpU4KpkQmyGLmeYyRGzyoqn8X7TItPPlMhcsY7JJjvLy8tC7d2+
MGDFC1PM6d+6M9PR07W3Xrl02ipCIHJEcVpwZMmXKFJw5cwYA0KRJE8TExEgbkNyIyZBNSaZMYU7y4uYG/N//Odd
csIzJZrXatGnTAACxsbGinqdUKuHv72+DiIhILhxpxZkYv//+O+bPnw8AcHd3x8aNG6FUKiWOSobEbOmhSaZ0tVV
fsACIjjbeOt3c5EWTnPXqpX6twu8ht7lgmZNNcmSuhIQEVK1aFRUqVED79u0xa9YsVDXQ0yI3Nxe5ubna+zk5OfY
Ik4ioiAcPHiAiIgKaHZ6mT5+OJk2aSByVRApv/aH5/+/MTNP2LdMQkyEbSqbc3GybvBhKzhYvls+Qp8w5dXLUpUs
X9O7dGzVq1EBycjImT56MF198ESdOnND719fs2bO1o1RERFKZNm0arly5AgBo27Ytxo8fL3FEEtG1OV5httqgVV8
yZY/kxZzNa8mqJN14NiYmxmgicuzYMbRs2VJ7PzY2FlFRUbh3757o90tPT0eNGjWwdetW9NDzD1jXyFFwcDA3niU
iu8rKysKIESPw888/49SpU6hdu7bUIdmfvk1ZC5Nqg1ZDG9mSQ7Bk41lJR45GjRqFt99+2+A5ISEhVnu/gIAA1Kh
RAxcvXtR7jlKp5Jw+EUnO19cXX3/9Na5cueKaiZGhrT8KE4T/Old3726/BEWuhWxkEkmTI19fX/j6+trt/W7fvo3
U1FQEcBkkEcmAQqFwzcQIML71R2GF+wsxYSErkM1S/pSUFCQmJiIlJQUqlQqJiYlITEzEgwcPtOfUr18fO3bsAKA
uZhw/fjwOHz6hDu3bUIdmfvk1ZC5Nqg1ZDG9mSQ7Bk41lJR45GjRqFt99+2+A5ISEhVnu/gIAA1KhRAxcvXtR7jlKp5Jw+EUnO19cXX3/9Na5cueKaiZGhrT8KE4T/Old3726/BEWuhWxkEkmTI19fX/j6+trt/W7fvo3U1FQEcBkkEcmAQqFwzcQIML71R2GF+wsxYSErkM1S/pSUFCQmJiIlJQUqlQqJiYlITEzEgwcPtOfUr18fO3bsAKAuZhw/fjwOHz6hDu3bUIdmfvk1ZC5Nqg1ZDG9mSQ7Bk41lJR45GjRqFt99+2+A5ISEhVnu/gIAA1KhRAxcvXtR7jlKp5Jw+EUnO19cXX3/9Na5cueKaiZGhrT8KE4T/Old3726/BEWuhWxkEkmTI19fX/j6+trt/W7fvo3

EpMO+ffuwcOFCAIBSqcTGjRvh7u4ucVQuRoa1KqSHzKYiOXJERFRMTk4OBgwYAE3VwcyZM9GwYUOJo3JBMqxVIQP
kUPz/LyZHRETFREdH4+rVqwCAF154AWPHjpU2IFelWTZvrFaFy+blQyZTkZxWIyIqJDc3F5cvXwYAlCtXDrGxsXB
zwL9sXQIbVZJEOHJERFSIUqnE3r178fnnn6NChQqoWbOm1CG5Nk2tiq4+R4sXO1ytCjkHLuU3gkv5iYgcAJfNk0h
cyk9EZCFBEKDQtyUESU8mtSrkHFhzREQuLzU1FS1atMD+/fulDoWIHIAskqOrV69i8ODBqFmzJjw9PVG7dm1MnTo
VeXl5Bp8nCAKAJiYmIQGBgIT09PdOjQAefOnbNT1EQkBwUFBRg4cCBOnjyJDh06YOvWrVKHREQSk0Vy9Pfff6OgoAA
rV67EuXPnsGjRIqxYsQIffvihwefNnzcCxcuxNKlS3Hs2DH4+/vjlVdewf379+0UORE5uuXLl2Pv3r0AgMDAQHT
u3FniiIhIarItyJ4/fz6++OILXLlyRefj4iCPgAgMDAQUVFRmDhxIgD1El0/Pz/MnTsXw4cPN+l9sJBN5LzOnz+P5s2
b49GjRwCA3bt345VXXpE4KiKyBkuzA5Kuu37IYYOdIlOzsbny4cnIyMjIy0LFjR+0xpVKJ9u3b49ChQ3qfl5ubi
cnCI3InI++fn56N+/vzYxGjVqFBMjIgIg0+To8uXLWLp0Kd966K/GOkq5GRAQDw8/MrctzPz0/7mC6zzZ8+Gj4+P9hY
cHGydoInIocyZMwdHjx4FANStWdz586VOCIichSSJkcxMTEQBMHg7dixYzrPSUpKQqdRndRecsOCOu8eQHy/ynBs3bqBz587o3bs3hwgwZYvQ9ii/
NNbZcd9KkScjOztbeUlNTzftwROSwTpw4gA2EASpUqhfXr18PHy/Px8A8OGH6J169Y
bN24gLCwMbdq0wapVqww+z9/fH4B6BCmZmZJUaTClMqlVAqlSZET0Ry9OjRI4SHhyM/Px8A8OGHH6J169Y
SR0VEjkTS5MjX1xe+vr4mnZuUlIQUvc6LqoEYqzDSR0cHn9CmREEFZUrvBvbYrpcBrDoWWnWYaMC8HSMLnk7xgQLB1Csb4k+j+vXVQJ22pdF1DVpt6gbFmJOLOV/gYCasNpxFPfQ0wL4S3nCoKRYlN3wEBVLXqDJO6oXH4MgECoXOQFd7oQB6vK4J8Nr6oAOnXO7RWPNUuMu9ssFBwUU/hGTNYs+UVVAkd2KDmaeeq+hm3EWuTzGD0hmcfw17rMGUfkcTXlwD6SJS+tKPGiOLQ/SUSvqiU8bTLlhhX4
QkccKdyI4f/48srOztffff/99PHr0CCNHjsTdu3fRqlUr7N69G+XLl7db7ETQgmjWIaFZmhstHC6TfaTWBefH8C/uBHJITQtEhQwJ6+UtlbHPmyZ/r8dXY8Q2fpqPUfxoOr9pQ1zy+N1jPXZ+W2XCQ1WNX2WEvoU2Fkz6tyE8O
OiIiI5Mcl+xwREZnqxx9/RHR0NB4/fix1KEQkA7KY7SR4tdzptdjMtetW7cwZMgQZCkZ8+4ZrwSUvsEf3fMb4TfgIBIe8PN89pB2
EQUBkZCQyMzMBALVr10blypUjoqIHB2TIyJyWps2bUjUJcXBwAdeuQ1atXswibiIxickRETik1NRFt94XHM/BzCTRG0ahYbqpKVGM+JL3rsCtNt2Q0XwZfsr0SqbDtcOC+sXOyrVf+PxyAr46VQvTJXbA
BBrBERBpMjojI6RQUFGDgwIHa1h7h4eHo2bOnxFERwwOSIip7Ns2TLs3bsXABAUFIT1TPP/9c4oiISE6YHBGRU/n
777/x/vvva++4/z/xOSpUkC4gIpIdJkdE5FQWL6s16TbWf0f//3f3j55ZcljoIsIIjqrIqIjqSxduhRBQUHYuhRBQUHYtm0b5sy
ZI3U4R4RCRD3D7ECG4fQiRPeXl53FSWyIVx+xAiomKYGBGRuZgcEZEHsLVy4EHzmrWegD79gdRBdsDKrqrbFfeTzmbMvFUZgW6ZCrvw/lzFS0DEx0hHU+9/+HvaOM19m79hdngZm6AsjnQTxeJgDT17pvAYQpNoCeOkz4WhB+fc3Hqfwzwmd4EVrtp41NG7MCQJCrfiW
t0Vo9Wo5BO8MtwJqlQBr1qC+lJHHMO3A8EI2FVa5MeOCZ3aHpKagAy2i3OWTVVLEnSsjBsIffhVIn1zN+b3uAHgtROIhF2ST0LijSZEjn7AYngBQ1ndubIMN05eCIC0ku04+wHeK+owazaUpCfHMvSrkJyqvNi4oVGuLZRqcUMiaV1aaWFyJgMnwgIwIIm2ZFFwR
gq2ilhUOY7FSTRD9r5Dp1lcbCqq9WhEQEZnYkXVAAXTMDLWIGPj0V1s7PLmwOBMXYzN3YfMR8AF8Z1pHeFRoPRXsUCSVzDvcNtD1l8T22Rh4NOEm9MShQf8/n0htg+u8zswnBiplcDUmNFXeAxkjoTMb6OXy75ORxUmMJ0KxYMXgI3wK0SgFFVyBjWJzDnMUZdqgQvakqNBJ4GyNWbNpCUF8l9ihtu3AgEekaxjFjQMrDkjYOHgnU7UmM4eRcGFE8yGxY
Eed0yRHq2I43Oqw1Nau11HZk3mbOuZk7xWyMhJBnLm05Wzi+k6fNWQWmBCFYwWSPk0FLTOBZ0s19oh/Ca96L1EETWZS0Lkyeq9TaYOU+Ok7jUxDmyH6Bc/8L3/Hj8CjoTNkK0/EWUe6yc5nXWpowK4qfhoj9kxXHdsxvSt4CqgCiYMjM17wsUqBEg3NG4t92ZEgKj

27dolnDt3ToiIiBAqVqwo3L59WxCE/zajbN26tfD7778Lf/75p1CnTh2hffv2QseOHYU///xT2L9/v1C5cmVhzpw
52vdt3769UK5cOWHMmDHC33//LXz11VeCl5eXsGrVKu05r7/+utCgQQNh//79QmJiotCpUyehTp06Ql5eniAI6o1
ty5QpI7z88svCsWPHhBMnTggNGjQQ+vXrp32NVatWCQEBAcL27duFK1euCNu3bxcqVaokxMbGCoLw3wbN9evXF37
66Sfh/PnzQq9evYQaNWoIT548EXJzc4XFixcL3t7eQnp6upCeni7cv39f5G+aiARBEJgcEZFRxpKj4r7++muhcuX
K2vuNGzcWYmJidJ6rSVru3r1r8DX79u0rtG3bVudjDx48EMqUKSNs2rRJeywvL08IDAwU5s2bV+R9/ve//2nPmT1
7tgBAuHz5svbY8OHDhU6dOmnvnt2/fXmjQoIFQUFCgPTZx4kShQYMGgiAIwoULFwQAAwsGDB7WPZ2VlCZ6ensLXX38
tCII6OQIgXLp0SXvOsmXLBD8/P+394OBgYfPmzUU+14wZDM4Q2bdoIgvBfcvTll19qHz937pwAQEhKStK+j4+Pj87
viIhMx2k1IrJYfHw8XnnlVSrVg3ly5dH//79cfv2bfzzzzzzz8AgNGjR2PmzJlo27Ytpk6Ytpk6ditOnT4t+j8TERLz00ks
6H7t8+TKePHmCtm3bao+VKVMGzz33Vq1SpyLDMzs8hzWrduDYVCob3fpk0bXLx
4ESqVCklJSShdujRatWqlfbxy5cqoV69ekff28vJuNGzbz0OWhd3Tb2DBAABghzBlStXEB4ejjNnzqBly5ZYsmS
JqPfx9PTU+5jwb/1P4QRGc7z4sTJlymj/W/WFrqHh2iND763rfTTP1bzf6tWrkZiYqL2dPT8nYsmS
cMRh/4ecTkXUwOSIiixw/fhz5+flYsGABWrdujbp16+LGjRslzOcgsDkZkZCTi4uIwbtw4rF69GgDg7u4OAFCpVAb
fp0mTJti7d6/Ox+rUqgQN3d3f8/vvv2mNPnjzAzM/mlbxBOXIkSN46qmn4OqqeffhfhrS5+fn444448/tI/
fvn0bFy5cMPm9/fz8UK1aNVy5cuV16tQpcqtZs6bJc6tfcDAgK073Pr1i2kpqZi8ODDBKFeunPY2cExMQixypVqoTatWsjPz8
fS5YsQbdu3XDw4EGsWLGiyHlRUVHo0qqUL6tati7t37+K3337TJg41AtAqHATz/9hK5du8LT0xPlypUr8f6TJk1xPlypUr8f6TJk1
C48aNMXLkSERGRsLd3R3x8fHo3bs3fH19MWLECePx+8BCDBw+2+LOnpqYiOjoaw4cPx59
//oklS5ZgwYIFAICnnnoK3bt3x9j3ChQ7Fy5UqUL18eH3zwAapVq4bu3bub/B4xMTFEYPXo0vL290a1r/B4xMTEYPXo0vL290aVLF+Tm5uL48eO
4e/cuoqOjXqNkJAQPHjwAHv37kXTpk3h5eUFLy8vsz4zkUuTtOKJiGQhIiJCAFDiFhERIERUXERIQiCICxcuFAICAgQPD0
9hU6dOgkbNmwoUmQ9atQooXbt2oJSqhVKkihIeHC1lZWdrXnz59uuDv7y+4oAFArta+qSkJAghIaGCkqlUqhQoYL
QqVMn7Xs8evRI+L//+zFdXB8F8Aj/q+z8qNRnZDqyVNJ4vjq/yr9FtLxwPkXGA2bCk2/8zPm55bk6lDc1Vgp4Mx
WFD744IMiBdp37twRwsPDBR8fH+13cOHBYPvs2PHDqH4/wVv2rRJaNasmeDu7i5UrFhReReOGFF4S4uDhBEP4ryD5
58qT2/Lt37woAhPj4eO2xyMhIoXLlygIAYerUqXq/SLSTyEIeibMiYgIHTp0QLNmzbB48WKpQyEiO2HNEREREUVEREREV
hTI6IiIiCuG0GhEREVEhHDkiIiIiKoTJEREREVEhTI6IiIiCmFyRERERFQIkyiMiIiKiQpgcERERERXC5IiIiIi
oECZHREREIUwOSIiIiq5P8BvycvW9i4688AAAAASUVORK5CYII=",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"# Scatterplot like before\n",
"plot_scatter(train[1], train[2])\n",
"# plt.title('Training data')\n",
"\n",
"# Calculate decision boundary (x,y)\n",
"x = np.arange(-5, 1, 0.1)\n",
"y = (b - W[0]*x) / W[1]\n",
"\n",
"# Plot the decision boundary\n",
"plt.plot(x,y, linestyle='--', linewidth=2, color='k')\n",
"plt.xlim(-5, 1)\n",
"plt.ylim(-2.2, 1)"
]
},
{
"cell_type": "code",
"execution_count": 40,
"metadata": {},
"outputs": [
{

"data": {
"text/plain": [
"(-2.2, 1.0)"
]
},
"execution_count": 40,
"metadata": {},
"output_type": "execute_result"
},
{
"data": {
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAkcAAAG2CAYAAAB1ZSLWAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnN
pb24zLjcuMSwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/bCgiHAAAACXBIWXMAAA9hAAAPYQGoP6dpAABnR0lEQVR
4nO3deVhUZfsH8O+IMIALCiiCkLjkUq5puRQpVi6+ilumpWLmz9TT3HMtsE1LfQ3M3DfcMnN
BRUQBF9BUwOH8/hhnGmCGmTOcmTNn5vu5Li6Ywzln7hmtuX2e+7kflSAIAoiIiIgIAFBO7gCIiIiIHAmTIyIiIiI
DTI6IiIiIDDA5IiIiIjLA5IiIiIjIAJMjIiIgNMjoiIiIgMMDkiIiIiMsDkiIiIiMgAkyMiIiIiA4pKjn7//Xd
069YNQUFBUBKUlU+P77781es3PnTrRs2RKenp6oU6cOFi1aaZPiAiYiISLEUlRz9888/aNasGb744guL9s/OzkaXL
MsLAwHD16FBMnTsSIESOQkJBg40oiJiIhqVRK3XhWpVJh06ZNeZP0uP31102eP/x4+WeXl5hjmMTAyOHuGvxXzcFrurp1
32iFKIiIiUprycgdgS3v37w1WnTp2KHOvcuTOWLl2KgoICuLu7l7gmLy8PeXl5+vsajQZ379+Hr554xGpTDkvd0hk/PXkohKa3jC9EqM67iH5+OeTbCOUYbyMrWRb
RlZ0gCLhz5w6CgoJQrpy4iTKnTo6ubW1sp1RvzCr/jYmtK6mPA6gjzStpBcaQdjAuDQIg4+E5pAZnuuQd2gHTa7/NqOF7b1GuZ6OTYl97jmf4iNbfr1i5EIjZSTqJYGgAO2xbdu5+6SWbPV/wQGiovN9sBHNQmnFxLeqHJ5Y/3d+jw6++X64Ys/T60nI5Ve8yYrzjzhfy/4qcKJd7oHNgLHQbCrCdVtbBPK/tn+iI/sXPp9iXndOXYf+PWXRJ6dNEC0
DRV3j1MkRgBKjPbpZRFC9D/VG0EhkBaz1a3c8JE83SEHUrWKVFwAt+KlkZOlS0RFd4oX2ZhPBe7/jzZ6A6QeX
auHbtWpFjmZmZKF++PPz8/Ixeky1SjLnALP4fE2mgEvP8jYh+PHY5OIkKVG5c6q4Q+OFMJNPvMwjBnYpRO1oAAMTy
Fjm3btg2tWrUyWm9ERERERjufu3bv4z3/+g1dffRVDhgzB/fv3bfp8/rO7mfIF+Ed4tRe5YzySrXeIYngGiBpKp9Z
nxPr3768/PyRWOFTUB+iqLmtVRIVFdOw3zPfbF3cuQPOWzTfD+bxt3t9mfHn+mZNvXGfbhZ+i44kzbaxaHnxm+pj6zX
XP9bvVBg0YMArVqxARkGgNsGP28bHj4vRWcf8OHeccXDKRqBjhGmBQtL3Y8UjL5JmqfnyeKHlVLwjFx+V65YNhN4
4S0jVH+WzZ2ZmHjrsjV0yOJ0IqfKqVDXNkEHWLcwEuR8ZrSq1ktS6cZ7r2dNMunKPFgAMGHhBmYuwqMu5wMHq5jQmhtMfp
xTrl0vVnffMWnWd2SBW64T0ftPx4uTVFJfcI6jnVYK4bgRqDvPRrB5t9UtGBP+BSqWUqXMVWR+EHHpXmW4NjuCI0xnB41
e69bzevr3nxWwUQC8wSgmdSCDo9nBOE3qtHcRUSPdd6ZlZc0JJo0fcJgJXYG3ElsHvBUBvbzGYXe6Zp2r3bbj9EyWcBVJYfWFbovDIO22nH3GggURsf2y9v1uq
PRaaSaDbnQ6B3QNixcrPpIU5WjhasVFuGxJxj+0iEjx0LBSBPiDzHh9gn1R6atNCICafQcqEPUnpp4Wd89PDyzMHMr12ic
MVyPPiIMFzsRcVT0pGkk2Ga6aDsuw1lm6VLXM4A9EfvmGxSP2wDUzSeyWGRYjlGBsMJfPe7kyqwcvuQPSKy5q7q15d0g
w1k2vvfQGlzFLq0yXZGNUkZtpQfF+SSNoOo0bsi0Csl3i3q90XKtGaq+zZabhoBxUVfD8u1KONY2FPVCppsOOHVaOZWXlTehOfhULs/mW1XrPxSVF9G8KnKs
Et7kyOjR6n7u2iMKQXv+ssKmtnmnQtjz/ITNpGLwfG5EvtqdUi95a43EQIrunM+BrqxNDR5ws2V3kzjmXJZhs47OUWBf2nK
rGNb/mZ1Id5QlY9pvpQ4WyGFt9sP31Lq2vPh91TDmc6GQwPktN9UHqnikjFJLZtmfWxjIYKmxAtsqjjNEPb8H2eR9FQm4cbHvq/ER2vA2XtOChDT+g2H
/tsocy4u9Ndr4ySJE64S1X/qLFa7vtmajLVtjt3k1+jQdQzkdCj/+oONqRjbqHDC1KiyDq6qCmr33SgG59s+6oEvTXVS
4XM7ES6CBgwxVoupfn2/YvjeO3xIKVd7O4uJ3KahMVcvEqZ6c4lqy5LXqiucQiq4bUtgUKY4s+G4VBWXWLTWYv9mJN2/hrbNG2DAkRQPxsVP2e3VRAivEOOd8pn
OYp4FmprCoVuxBvz3QkRlmTLQ7mrY9mhoXTvsejfsZnO25OHFQ6cTz+nJVhTRmiDOu/hyb+7pmvq

o3bs3xowZgyeffBIZGRnIyMhA7969bfqaiIjI/m7fvq1v9OgMOK0mF40GGDkSEISSvxMEQKUCYmOBiAibTbHVq1d
PXzRX3Jo1a6BSqbBkyRJ4enriiSeeQHp6OgYPHlzi3OnTp6N9+/YAgPfffx+vvPIKHjx4AC8vL1SsWBHly5fnlB0
RkZMSBAFDhgzB8ePHsXr1ajz99NNyh1RmHDmSy65dJUeMDAkCkJamPc9GWrVqZfJ3Z86cQdOmTeHp6ak/piuwK86
w2C4wMBAAkJmZKVGURETkyNavX49vv/0WZ86cwauvvvqIjWXNYXIkl4wMac+zQoUKFUz+ThAEqSeseMcXd31/+
su6awsFCCCImIyJGlp6fjnXfe0T+eP38+vLy8ZIxIGkyO5PJohEWy8yTWsGFHD9+HHl5efpjhw4dEn0fDw8PaDQ
aKUMjIiIHIAgC3nrrLX2vu759++KNN96QKSS1gYYEBysrS0yRqUCQkK058mgX79+KCwsxJAhQ3D69Gls3bp
Vv2Fg8RGl0oSUhIs4ukmwREZFyLVq0CNu2bQMAABAUF4YsvvpA5Iukw0ZKLmxswb5725+LJhu5xfLz
d+x3pVK5cGT/99BNSUlLQvHlzfPDBB/oNBB3rkMzp0aMHunTpgpvvDwcFSrVg3r1q2zVchERGQnZ8+exdixY/WPly1
bBl9fDxDTPPPcWVmiCJKVtRLOMX/k3YBXMkuckmB27cTqCpjrVUMhh7ZlDwObn7vQWOdrDbR3n9n07Wq6NGa4rKoLVkOgtLtK2AtxpuKzJ3PaV14SGpYgYPKLP0d8ET6m0wwJURmkKIBbzNSZL
sQkO+SYfbo9ExR4zZ3oENrX0z1CYGHyCGO+JZL1JyXCXLM58EMFn7iVaKKUMCybJ2vz4NmJJXpSK3SxP8Ql31kBKm8
QL9vjeeEAcvqC7wdzg/MQdqd5ISBDNLnROLnx0C2bZhVFRSGbD4ufSJoJzCGmXQWMKfsixcHULOfR9RSGIjK3fb4k
4uBnm2sqXycQsSnhbMh8dlRyW5wGQjJt0XQi3uO4c6dk4vqyr2qQhQ2OJMkbgE8EqaGv0dxSURd5BURJrAB1rcOmTRIEfGH/CLcSFknw1lAp
BP0RVqaymsabJ6Ss+TMhnoA/0gsGtn6nG+sHCPP2h/FCT3MEZg4bxb4LwDVrFpFeuD9XGmhgu4FoohYiBTc3u1XZF1vqb3JMXSXfo7lWuO5BCsSC5Te8j7/QeSlsx3+gEVgF0k+w8d6NoNZMwYB4Ie+LbXZv+SCRc4kDTP0x3rw8bHRR5NWYR5Wg9lbYbTGBrVu
qhMfVQ7yC3QxTD1yUeJPvCC+0ZD5LebM8ab1UcvvN47Oq3WCaCgSvn5ui7+iuH8dzU62Y3r0crsgGzmeMtFrWbNRM+0Ntd/9NGKHt+0Tc8BE6oUj5nlXJgKxc38QzHkwCdzMI+sgQzvKYhDg0wk9p+J2MjJvVfdC0iA+g0fSr2+mjS
3sWhQIlUl+BfvcnCEVxXSYINrQTDCNvDcTn4gP/o3oZpGwwNxSPgpeI+2jUCv9MrEt+Oi/RLRjUkY4v5+LQg6f8yZHm7f4AMg6krVVtV8wFkbDqHpYwT4IB1YRnSKRxd+cEIyVlfK6GjRiQ+6GCVCDw1tPoyArUHf6OgNn1Y+FJPcmSUQbxNrOVtF3RxSPD/7pbN8M3QLm6vzFNKnNnhTF9N/XkJrBpuVvfqyhpD
Pq+YRKbaHFpXw7kSWP6s6NAbCZKm0tvvAr6Z+ybbpFQm8tGnGSNGRG5BENoHJAB1YuDodBtYJp/Rcit3tyzM3APrxiBNcGG+RTFxqn9NhCvXARWSvDNahqU/cW8z4bMdPM9Cqa3Zlw5nbkckgsY/RuN6pGsiA+UKC7MrbYSQ+eswHK/g/

a7f/+7//w5JNPyhwVGVLcarXi+8sIglDqnjPGzjd2XGfGjBmYOnVqGaMkIrKtyEggIkJ5rQgIGDVqFC5dugQAaN+
+PWJjY+UNiEpQTHLk7+8PNze3EqNEmZmZJvedqVGjhtHzy5cvDz8/P6PXTJgwAaNHj9Y/zs3NRUhISBmjJyKSnhJ
bEbi6H374AcuWLQMAVKxYEStWrEC5coqZxHEZivkT8fDwQMuWLbF9+/Yix7dv34527doZvaZt27Ylzt+2bRtatWp
lcm5XrVajcuXKRb6IiIjKKisrC0OGDNE/njdvHkJDQ+ULiExSTHIEAKNHj8bXX3+NZcuW4fTp0xg1ahQuX76MmJg
YANpRn/79++vPj4mwaVLlzB69GicPn0ay5Ytw9KlSzF27Fi5XgIREbmomJgYZGZmAgC6deuGGQMHyhwRmaKYaTV
Au+/MjRs38NFHHyEjIwONGzfGli1bUKtWLQBARkZGkZ5HtWvXxpYztzYtWzBq1CgsWLAAAQUFB+Pzzz7nLMRER2d1bb72
F3bt3Q6PRYMmSJaXWy5K8FNXnSA7sc0RERFLJzs7G2bNn0bZtW7lDcXrcW42IiEgB/P394e/vL3cYIaiao6IiI
U5PTp09wWRIGYHBEREdnAX3/9haeeeegqvv/46rl+/Lnc4JAKTIyIiokVFBQgKioKDx48wI8//qjfKoRGckRERG
RxGbMmIFDhw4BABo0aIDJkyfLHBGJweSIIiIhIQocOHcK0adMAAG5ubli1ahU7vriaeWq1EMTPhcvUcyD7pSuTPVvI4kRbhMA67vNBkIG5YMbNRMgA4yjHA1pBiSACAOONUqITBWh/yFrFTnJE20iJEA0Ok/FyLA1tIL
VNH6hCVPhOghIrcHgOqBOcAS9iAcxFxLXv1/6ErZChS7QdtnAvtOxffkeSG6Rm7Q1j71b+UZTAZLVNfa27tYdyatOqj4aGzJnYn3mXbQezalMl0D+kOlEBmDLgEW+1dEz7p+
XQ+mPKq0RXxuIx7zB1yvbAgVvh5BpehHcQjq+6eWmfLrLFc8cG4mc7Pz1fGI6HbTxrt4+sMH0DKdDzIeUeHN7z8mjXkKPNGCLDHyVfgtVj9vBYAlmhMAXSzHUROOyRkrwK/F
1BpyQoXJAXKO0aqJqDEyXPvmwyTL3FEBedpbmvKDw2+SPOwMB0PGlqv2aAnd09u5Fvxo9aG8b2hzxO5NdzjVpwxxYzS3u20Hdxg3x9/dHYk/+yMxe7LCk2n7k1jfmHiPdXt9
mjOLKT0lPvNvNdEr2t1q+c6IG1GwfNxlgj+IE8/rOrvl9/HyxNn8P5WsnlILXUXCnhPvMpEt8Jx4+KPrHrZeG7MM++cvvQ/r1+Y0iepZk9s/8SOl2uJoHq/VSfpk1WE9Sy3nr
i21Uz8SE+L4TiP5e42UlNjEtJDlMS0LzvXmfvqk/D6Wc3DN2bPdb//JI1H0ZXE3Id/3zPRegpmEt5Zy3JQvZJYqq+n2iHYwRVVGmq3/OnLW/Hg9Jq7/kM/HEd4/+OZhV+gjSz
YOX/mj45Jc+ZMPOyc8xa/zerDB6K7z4tEEUtV43/lRmzf7eWRfUGPVhx6q7ihLCZgrY6G+mSX3WxVfuaSTdAb9HzT2mtZJ8hwhgcWgfxp7YpQZVxohKoqdpPxDN+fuZgqAWt
VpWXhGlERQlYWGQLRELyWmaYwVG5rkxYwGr5VGrV9Tp0qWexERLYWGQlERGhXpWV

kaGuMwsJKHzHSaMSdryQFBQWIiorCkSNHsHXrVvz666+oX7++3GGRgohOjvz8/IxOnfn4+KBq1aqi7hUXF2c2ETl
48CAA48Oh5uaPe/furf+5cePGaNWqFWrVqoXNmzcj0sQ/pyZMmIDRo0frH+fm5iIkJKTUGImI5ObmZvly/cREYOT
IovuzBQdrp+icYaTp448/xpEjRwBoGxfz/+Eklujk6MMPP8To0aOxatUqBD5aAnHt2jWMGzcOkyZNEnWv4cOHo0+
fPqWeExoaiuPHj+P69eslfpeVlYWAgACLny8wMBC1atXC2VJ66qvVapOjSKRESpeYqK1RKl7AnZ6uPKm0qbDBw5
g+vTpAIDy5ctj9erV8PLykjkqUhrRydGXX36Jc+fVWvtbMgPBv4OoVatWvruopcvX4dzZarUlQHmxWVha++uor/bm6zN0Uf39/+Pv7m33
Otm3bIicnBwcOHMMAzzzzwDANi/fz9ycnLQrl07i2O/ceMG0tLS9EkdEZFiSDAPptFoR4yMrWWwWMmQIXn311SIr1Ro2bIgZM2a
ge/fuuHv3LuLi4tCjRw8EBgbi4sWLmDhxIvz9/dG9e3e7vwYiqtJNA+2a1fRWxSn9I7aEyZMMwJkzzZwAAATz/9tL5
RMZFYopOjKaX1qrehNWvWYMSIEejUqRMA4LXXXsMXX3xR5JwzZ84gJycHgJycHgLa9wIkTJ7Bq1SrcvnbgYGBCA8Px4Y
NG1CpUiW7x09EZBUJ58Es7ZStxI7aO3bswOff/peRzrO1iiRTSCJSDa6Lo+mhntEdlMTgbCw80/bVKSskaObt++jSZNmuDKo/dp3rx53M6KyvT5LbrPUWpqKl555RVUqFBBBv0KtatWqqFK
liujVakREVAox82AWCAvT5lKmFvmqVEBIiPY8JVm0aJE+MerYsSOGDx8uc0SkdKKn1d58800AwLx6bLHJ1yd58800AwLJlyxAQEMCOo0R
EtiLxPJizdtR+77334OXlhU8++QQrVqxAuXKi/91VITOabWKFSvi8OHDpW7Z4ZkUwrUZEsrHRPJix+u6QEG1ipOR
l/Pfv3+eyfdKz695qTz/9NLLS0lwmOSIiko1uHsz6YSoSpHEyEyMSCqik6Ovv/4aMTEx9KbVmG
atKlkwRERuTQbzoOJ6ajtiL799lsEBATg+eeflzsUeAwcO1B9TqVT6Crt3Cup1ljfU
5suM8mCPtxZaamoq33noL//zD0aPHo1Zs2axzogYI+3FVlhYiOjoaNy
9excAcOvWLSZGJDnRBdkVKVKEIKlTAswPHUK9ePVvF5FBYKe1E1ESiL1CI+pHps6fxfbey+2OXmYyYyYMQCAWrVq4x/x
/Mxll1z5HHTt2xLFjx8RbKeRkERdkVKlTAsWPHUHOUnOHCmq12mjSpHRMjohacvhNRjuSAPJkBJJRHmq64xpn
mRE+rdevWzBicAzACPemTEgXZr732mxwhWi+KSIHrJlJrKbV+Hy/EREfPgtNGnB99VJm5dkQbyzd
rlTa364wF2ZxWIyJHJ/EulSHoNxZUxbJd3eiPuEw+Rn0Lk1ZWzo+7Y1jdxYVeqQMmRdgSeewOHDh+Hp6vVFs+u0WmhcK
vZ0uMiIiUQOJdRvQCA6U9z1r79+Z/H//3f/wEAypUrhx9++AFqtVqfzO/sbPxTc6GG6jmNBMVSa
nRwsXLhQ8PDyEmJgYdq12mjSpHRMjohch7lkxdqv4omJTlKSZdcnJUkTvy7Z+vZb4wlgJBI
EDVRCoaUvwAYyMjIEPz8/AYAAQFi1apXNn9RnNkRmzH103SK8vnt+hpNQDYtGkTPvvsM319UhESIvZX
D47QaketITtZuuWEr1appp7w8PLSPpW6yaGn81aoBWVlFj5WDhcRipq4YnlTrFOkraYGRIEAa+99hp+/vlvvnAEC
PHj3w7bff2naDc10t10CTZleXz26qy/+7du6N79+WEpE5LDE9gESKytL+zm7aJG29kfqjL+zm7aJG29kfqJfiWxl88MQKAMOxCCCz
rJJl4s0OZC8iNWbZsmT4AxCggIwJdffnbxAhwrD1SyGGIXsp/8O0NB7N+/v8FhQ5QGUOhEkSZ1tNo9fvtp2d
Am5j07KktnpZZ6CX5Z4g+EZZnVgR8y0LNnyXxCt4ebtZvcpqamIjY2VzyJlqFatnmU3E8NWHTRJ0UQnR+++y7
S0tJKHE9PT7774rSVBERKZIvfO8IXPjipR0eYCUTRYtSbZM5RsZsCyzmv1NoNFWAbpjsbHik1WNRoPo6Gjcx
XADBo0CB069ZN3E2s5Sh7pJBDEX0b9fqd/sL/yMHFV/+62/F+NMrH3fnpHfW6PA4I6Fb7hP6pMnP9rb2d34WKNcZmKNPH6aw/G2Ycnop9yZF8V9Pj
NlE0WLYl/wQLjCdQuhCENwSiE6czqQbUQJGabHsaydg+3+Ph4/P777P777wCA0NBqzJkR9wNysIR9kghhyM6OVKr1bh
+/XqJ4Ze+GDdJr+zD1UqlN9QobsDuXwuphp8k+k8S8fk0HyIswZAjlvEPhAx96QGSpJHXDs4qRCmo5QrQ0yCfPz3a4b3vAc53+XbyZqzGM+Ou2uXLu36F57mpqQ
mkq1evYwnUIVwQyy0vxCKJ0iPTjz4ZjwKYX4YS8wM1N9//63fEkSlUmHjLG8azgC2QvSrJ8YXPlk1hAaXlxvMpGjRa+nhgC1BbYekx4Sw7qOSq97fp7Lmfj9n9kM47kmRu7XNe0BD97hc8V5HKwaUgMhO
bt2/rj926dUto37690KtXtZdHL1Hl5hZzkGw7+WPzDPdBkOzX7Qpc9ViUjqW4wIKZpxLOnLPnKOMu2o42ciO3zbNh0bjZ0p1uVE
FfhESIggJCTb5MygoKBCmT58ulC9ulC9fXhgzZow0LR84axt6QR6+blMmuS/nT09Px/PPP48P2jRogUBEAdi
+fTtCQkJskjMLJh0v5iRyD1MvebSk/XztSY2xVGGC71eFil9cbnn/2rLbDd3q6dll/GHahsW8G3hgZiOc/CNMv3w8
N1Z5j7JOjLK/r+PHjqF+/Pw9PUs/UYq9Vkxhh2ynYtftQwDgn3/+wZo1ia3DcXeyuE
xOSJyDJb28ElKcowV16b2OrNmWxBLn8/anEHMFWkBBtCYvUiZG1OYM
1PRDL+rru3buH06dPLkug5NLK8vktuiCbiEgOtqiba9ZFSbC1bnSqBNaSJyNUVS12iA+fMt74G
oU9YC8vHjx6N169169aYPHZkNGHKoOB9BkEgm8ixHVSoyC/Fa9daH6Ohbdu
26fdN8/LyEs6ePWPWvyyyfH5z7T0ORKUpkJBARUba6WVPTULp+SZauroi9hxTPVIaKXog3rp
1CwMHDtQ/nj17Ntq0aWPR3F27NkrPSdHQ/nj17NurVqydYK5Vq5jVVkY5vkGAPDSSy/h19
/RblyFlZ3WFt0ZsvVbeTQ7FpzVKdOHdy4caPE8du6bNJC/t/4ZcTPsuOdlDhxPaYw1DZ5NxVIaKXog3rp
3+sTIx8cHy5YtAqwrOrN1S/XS2GoDQLIL0cnRxRxYsXl4f09HpqPMcu4TB9aBNVk11DjXfe
WbGFiSQ5w7do1xMTC8cUAXwrOzC4pqjH3/8Uf/z1q3o3+s0WiwY8Ohbjo8Ohla
GShocEZHUpNhhnVOw9bL23Z23qS5nMDZ7VHx5vaXPMcu8N//lj5iZMMmlSAIePvt8Cbb75pWDGWFp0Jma
4TsrGWLYqLiO7srjmSDf8qVKpUPwSd3d3hIaG4rqfqfZQyYs0RkXORol+S2HvYq0eTJXXHUsViaW+lr7/
+GoMHDwYABQE4OTJk/D397f4XZ2UjcjDRTA+OTc1BCc4hnXbp0+dk1R3HbtLyXeRQKlCHE3Uo9HyLgkSX5ZNzJ3xTlqUCTJEjB5Hyn

6JYm9h732NnVz0w6C9O2r/W7sflLEImbG6umnn0aTJk0AaBMluyRGgO2H64xhLyanIbrmKDU1tcRf7tu3b0sVDxG
RzYkpXZHqHlI8p1TKGouYHKBZs2Y4ePAgvvvuO/v+A1pspboUbFlcRnYlein/zJkzERoait69ewMAevXqhYSEBAQ
GBmLLli1o1qyZTQKVC6fViJyXFO1vyrLZq9wtd6yNRTGbANt7IzilbQDo5Oy6t1qdOnXwzTffoF27dti+fTveeOM
NbNiwARs3bsTly5exbds2UQE4OiZHRPJzpITC0cjx3pjPAdIBVENSkof8OYC9NrgDlLkBoBMry+e36A7ZGRkZCAk
JAQD8/PPPeOONN9CpUyeEhoaidevWYm9HRFQq9vArSZcQ/fAD8M03QHb2v7+zx3ujm7EyngPkA3gV7u4q+PquBvC
k7QKxhBQt1S2lK+jq2VObCBkbrZKiuIxsTnTNUdWqVZGWlgYA+PXXX/Hiiy8C0C7XNNb/iIjIWnL28HNUhi104uO
LJkaAfd6b0oq6gY8ApKCg4CjeemtAidXNdqVrwrRxo/bxG2+YrlSXiiMVl5HVRE+rDR8+HD///DMef/xxHD16FBc
vXkTFihWxYcMGzJw5E0eOHLFVrLLgtBqRPLgquiRL90Sz13tTclRvH4BnARSifPnyOHDgAFq0aGG7AMQFZ98hR84
Fy86u24fMnTsX//3vf/HEE09g+/btqFixIgDtdNs777wj9nZEREZxVXRRpS2fL85e701kJHDxora+eNmyfxAc3B9
AIQAgLi5O3sRI7iFHS/oqkMMSVXNUUFCAIUOGYNKkSSX2UYuNjZUyLiJycXKvina0f/iL2RNNxx4rxnU5wHffjce
VK2cBAK1bt8b48eNt/+TGmGvCpFJpmzBFRDBhIZNEjRy5u7tj06ZNtoqFiEhPjh5+OnJtjVXaPmXWJDq2eG+M2b5
9OxYsWAAA8PLywqpVq1C+vOj1PuIZe8M45EgSED2t1r17d3z//fc2CKV006dPR7t27eDt7Y0qVapYdI0gCIiLi0N
QUBC8vLzQoUMHnDp1yraBEpEk5OjhB8g3I2MuIROT6NjqvTHm1q1bGDhwoP7x7MGDUf/wYdvvRG/qDfvhB8uuZyN
GKoXoguzp06fj008/xQsvvICWLVuiQoUKRX4/YsQISQPUmTJlCqpUqYIrV65g6dKlFnXlnjlzJqZPn44VK1agfv3
6+Pjjj/H777/jzJkzqFSpkkXPy4JsIvnYu4efXEXgluxTFhFRegud4tfZa2HUm2++ibVr1wIAXlSrsTUv799/ddu
qALq0N8zSjzQ2YnR6dm0CWbt2bdM3U6lw4cIFUQGItWLFCsTGxppNjgRBQFBQEGJjY/Vz33l5eQgICMDMmTMxdOh
Qi56PyRGRvOzZw8/SBsf/+582OZKiHklMQvbDD8aTRUOWvjdS1FQVFhbiww8/xMyZM1G5sBAnAAQXDx6QNlMz94Y
B2hdSWMhGjC7Ork0gU1NTxV4ii9TUVFy7dg2dOnXSH1Or1Wjfvj327NljMjnKy8tDXl6e/nFubq7NYyUi0+zZw8/
SmZY33gBu3vz3cVkGSMSUyOha6BRPFqtVA958U/s+WfLelGWVe9GkqhymxU3Dq19/jeysrKKJkS54cwXQYrM0Syr
TddN5bMRIVrJDxZw8rl27BgAICAgocjwgIACXLl0yed2MGTMwdepUm8ZGROLoVkTZmqV1PYaJEfBvPZI1AyRiV+W
VNVk0NSNlyWswllT19N+Fb7OzTD+hYXZX/A/RmizN0jcsNlb7Yorf2xZDjuR0LEqORo8ejWnTpqFChQoYPXp0qef
OmTPH4iePi4szm4gcPHgQrVq1sviexamKVXMKglDimKEJEyYUeY25ubn67VKIyLmVvi2GaWVZIW7Nqjxrk8WyrHI
vmlTdAaCt23TPtrLngrVZ2tmzlj1fRATw6aeO1Y+BFMOi5Ojo0aMoKCjQ/2xKaUmHMcOHD0efPn1KPSc0NFTUPXV
q1KgBQDuCFGjwf5XMzMwSo0mG1Go11Gq1Vc9JRMpW2tZY5pQ2QFIacwmZrkRGipVnYqbwDF9D0aTqAoCnAYwHMAZ
XYUV2Z22WptEAS5aYfy7dG2avIUdyOhYlR0lJSbhw4QJ8fHyQlJQk2ZP7+/vD399fsvsZql27NmrUqIHt27fru7T
m5+dj586dmDlzpk2ek4iUz1Rdj58fcOOG+evFrhC3516l1jbW/Dep0gAYAOAmtMmRgF0YizQEoybSUQ4WZnfWZmm
WdsIcPJgjRFQmFvc5evzxx5GV9e+8cu/evXH9+nWbBGXM5cuXkZKSgsuXL0Oj0SAlJQUpKSm4e/eu/eu/pyGDRvqm1S
qVCrExsbi//7v/7Bp0yacPHkS0dHR8PB2Pb2Rr9+/ewWNxEpj+G2GGvXar9v2GDZtdY0XrTXXqXWNtb8N1maA+CPRz/
XBvAOCuGGkdDuQiug2OyBqezO2izN0usef9yy84hMESykUqmE69ev6x9XrFhROH/+vKWXl9mAAQMEACW+kpKS9Oc
AEJYvX65/XFhYKEyZMkWoUaOGoFarheeff144eNyckRAAg5OTkSvRIiUqKHDwUhOFgQVCpB0A5tFP1SqQQ
hJER7XlmeIylJENau1X4vy71M3d+a15CUJAjAcQHwePT/XpUA/F7k2u5IEO77Bxe5YWFwiCAkJJQMRHtD818G/38
v03VSsvUfEkmmLJ/fFvc5KleuHK5du4bq1asDACpVqoRjx46V2GPN2bDPERHpSN2UUo7926x5Dffv58PH5xkUFBx
7dGQcgFlFrvX1BbzVGtS5uguByEAGApFaMwxzP3cr+Z7oehWZK7Qq3ovI2uukUpYeCGR3Zfr8tjSLKleunJCZmal
/XLFiReHChQuiszGl4cgRERlKSNCOvhgOVISYGCARe5/gYPH3sYbY1zBx4kSDEfvGAnC/yGiTqQEclUr7ZfS+CQn
/nmDxRWW4rqx0zyvqRZKc7DZy1LVrV/1Krp9++gkdO3YssX1Ioq13ZrQzjhwRUXFlHfGxZLsQWw9EWPoa9u7di+e
eew6Fh YUoX94dvr4HkJnZXP/74GDg/n3TxeqlDuZY2/7cnm3TAfn2laEyscv2IYYbC5Zm+fLlogJwdEyOiEhKSvq
c/eeff9C8eXOcO3cOgHRzfHjJxZJqjQa4MUXzd/L5FZm1maa9pyTtHRfGe7X5lDssn2IsyU9RERysHYVuxyuXr2
KcuW0i5rbtGmD9957r0TroHXrLLuXyYVmpfUiKi0BsmcPI2tX15FiWbyUn4iIyk5Jn7OPP/44jh49itjYWKxatQ
rly5f897S17QHMSkzUDrGFhwP9+mm/h4Zqj9ubzV4kOSqLp9VcFafViEhKzjZDY5MFZI5QlGVI7lVyZJWyfH5z5Ii
IyI5024WY2m1JpdLWFkuxXYi1CgsLLT5X1+EbKPmarOrwbW5rEUC7tYhGY3GMZSb5iyRHx+SIiMiOHP1zduPGjQg
LC9MXYVtC0g7fYoqy7MlebczJIXBazQxOqxGRLdh7NbpRxQqeM+rVQ+NmzXDz5k14e3vj+PHjQ5lGVI7lVyZJWyfH5z5Ii
12hojc9auBfr2FXlzCdhjlZwc3UGdkF1WqxERkXQiI7Wbzsv2GVgsOxMADPL0xM0HDwAAr7zyiugdECRZQOboxc+
2XiXHLtwOgSNHZnDkiIicjpGC58UAhj76uUaVKjh57hz8/Pwsv6dUox2uXPzsaIXoCseCbCIisoyRgufzAEYbnLK
0fHn4Vali+T2lXHbv6EVZtuKIhegujMkREZGNaDTapfvr1mm/O8TnWrGCZw2AAQD+efR4CICXs7MtL3jWjXYUL6J
OT9cetyZBkqL42SHf/FI4aiG6i2LNERGRDThs6Uix7pKfAdj96Oc6jx4bO88oc6MdKpV2tCMiwrotQc6fB/bsET9
V57BvfimU1B3UBXDkiIhIYrYYTLGKsdETg0Lm4wAmPfpZBWAlgIq6X1pS8CzlaIexqbm6dYGbN7Wr0jp0sDwxcog
3XyRHL0R3MUyOiIgkJHfpiC4f2jUqEQ8CQ0vWAWVl6btQ7gCQ/+i6cQCeA8R1oZRqtEOqhEbuN78slNAd1IUwOSI
ikpCcpSO6wZfPwxPxbHxPeGQZSTZ699b3Bxr1KEF6GcBHgPiCZylGO6RMaJRct+OqhegOiskREZGE5Cod0Q2+XL2
iwTyMBCCU/B+8LtlYvx7YuBGoWRMdAWwGoAbEd3uWYrRDyoRG6XU77MLtMFiQTUQkITlKRwwHX57HLoTAgmTD3x+
4eLFsvYl0ox09e2oTIcPRH0tHO6RMaJyhbkf27qAEMDkiIpKUbjDFXA9DKUtHDAdfAlF6EjEG2lVp71y9CpUU3Z5
1ox3GVodZsheKlAmNHG++Ldi6CzeZxWk1IiIJyVE6YjiokgHTScQvAOYAGA6gx6JFkGyDhMhI7ShUUpJ2z7OkJG0
Ha0umgSyZmgsO1g6PmetZxLodkgiTIyIiidm7dMRwUGUXwpCGYBSiaHJwE8Agg8ed+vSBylRCYg3daIeYZfe660p
LaAQBuH8fePFFy7pvs26HJMC91czg3mpEZC0pN1cv7V7FtyPrjkR8h54AgHLQ/i++L4D1j+7VuXlz/HLkiLTJUVk
Za9zo5wfcuFHyXEv2GuPO9i6vLJ/fTI7MYHJERHKzpOGzbrUa8G+CNA8jEYIrWA9tcgQAVStWxIm//kLN4iMrjsA

```
woaleHRgwQJvxGSN2A1omSy6HG88SETkpS/sjFp9N2oRIhOIiuvl+i6HqCvrrFi5Z4piJEVB0as7NzXRiBJS9+7Y
lG+MqbX82kgyTIyIiByW2P2Lxuuj//VYOBU8vRW6edlvZ3r17o0+fPnaJvczk6r6tS4hGjdKOMIlNqMgpMDkiInJ
Q1vRHNBx8+fvvxdi69VcAQGBgIBYsWGDbgKUkR/dtwxGm+HjtViuGHH1/NpIMkyMiIgejG7yYNcuy840Nnmg0Gnz
++ef6x0uXLoWfn580AdqDvbtvmxphKn4+4Lj7s5Fk2ASSiMiBGCu+Nkc/eGJQdOwWGIg9u3YhdswYqNVqdO3aVZL
47FbXbM/u2+npwPvvGx9hKs4woWKjRqfF5IiIqBT2XOSkG7wQs4ZYP3hiJKvyCQ7G8nnzoImIkCw+c6vmJGVN923
DP7Dr1y17nqwscdko4Lj7s5EkmBwREZlgz2SgtPKY0sTHA24/mMiqHtXIuEnQ/NBU4qYrw7FZf0Uxe40Z+wNzczM
9BaZrB1Ctmvi4HHl/Nioz9jkyg32OiFyTqWTAkv6D5hgbjdq1S1sHLEZsLDD300cdIK9cQR6AUQAmAAgxDFhMPyA
T8T56CqMkeIqyEzvsZvgH6etr+ZvvEC+WLME+R0REEhK7yEkMUy13fvhB/L0iIlCk6DgOwJcAmgD4yTBgS/sBmWD
Nqjm7smbYzXA7EXPF3zrcn81lMDkiIirGVslAaS134uMtv0+RhVqPal92A9AtbrsHoFbxi8zUyJTW71CqlkM2Y+4
PzFBsbMmNcUvb380Q92dzGUyOiIiKsUUyYMlolJubFYMXgYG4C2AAgMJH50wD0LT4haXUyJhrIC1FyyGbEvMHkZB
gvGbJ1Ia11aoZT6jIqbEgm4ioGFskA5YMbuhGa4qvXDdUYqFWWBjGVaiA8/9ou2C3AzDW8AJdjYyJfkCWFFpHRGh
vodvYtjgzT2F7Yv4gSluGL6b4m5waR46IiIqRov9gcZYObsTGihu8+HX7dix6lBh5A1gJQP9RbqZGxtLaKsD0rJN
DlOHo/sAsVdofhmGL8Q4dmBi5KCZHRETFlFaCYm0yYOngRkRE0f3RkpK0n+Vz55b8rL558ybeeust/ePPqlRBPcO
bmamREVNbZWrWySHKcAz/wCzBZfhkBqfViIiMsKb/YGl0gxuWTE3pBi/Meffdd5HxaBSkc+fOGPrzz8Aff1g8JSS
2tkqWWSdLu3BGRgIbN2pHfMz1NZJt/o+UgskREZEJUiYDUuyGYejAgQNYv349AKBq1apYtmwZVOXLi9rSwpraKks
TN0mI7cLZq5f2zezVq+TvHGL+j5SC02pERKWQsgRFyqmpZ555Bj/++COqV6+OL7/8EkFBQZZf/Gjd/vPp69DTPxl
uMD7SYk1tlWRK63vQs+e/S+mK69lTuyKteA2SQ8z/kVKwQ7YZ7JBNRFKTcr+23Nxccf9vMjIak4ZgxGIeEvFv4iB
FJ3A9sS9YipbcttoUz56b7VGZlOXzm8mRGUyOiEgWtvgQNrFuX4AKAoCe+A6bHiVIISHW1VYZfU6xG9QlJ1u2nUd
SkmVzfFK9l3bfeZfKoiyf36w5IiJyNCY+hM+9/z52V6yI/v37Q2WuW2RxpazbV0EAVCqs9Y/F93MjUKOmmzQDItb
uVitlF06pEhrZdt4lOXDkyAyOHBGRXWdSTHwIawCEAdgL4PXXX8fSpUvh6+tr+X2lHo0xpyxTY1LFKtXuwYrYeZe
K48azREQ2Ym5rDUmVMrozG9rECABOnDgBDw8Pcfe29wZpZdmgToounFLuHuzwO++S1JgcERGZYO2CKauZ+BA+BmD
yo5/LAVg5ciQqVqwo7t6Wrts/e1bcfU0pSzImRRdOKRMah995l6SmmORo+vTpaNeuHby9vVGlShWLromOjoZKpSr
y1aZNG9sGSkROQcqBB4sZ+XDNAxAFoODR4/cAPOvvL/7e5kZjdKZMkSbrK+sGdWXteyBlQuPwO++S1BSTHOXn56N
Xr14YNmyYqOu6dOmCjIwM/deWLVtsFCERORNZZlKMfLhOBnDi0c9NAcSZOM8sS7fYUKmkyfqkmBqLjCy5l0rxzeV
MkTKhscVme+TQFJMcTZ06FaNGjUKTJk1EXadWq1GjRg39l6gCRiJyWbLMpBT7EP4D2lojAPAAsBqAuiwfwpGRQFx
c6edIlfVJtUGdtV04pUxobLHZHjk0xSRH1kpOTkb16tVRv359DB48GJmZmaWen5eXh9zc3CJfROR6ZJlJMfgQvgt
gAADdrN5HAJqqVGX/EH78ccvOkyLrk3O3WqkTGofeeZek5tTJUdeuXbFmzRr89ttv+Oyzz3Dw4EF07NgReXl5Jq+
ZMWMGfHx89F8hISF2jJiIHIVsMymPPoSnVqyIC48OPQtgrFQfwvbO+soyNSbFc0uZ0Mj5WsiuZO1zFBcXh6lTp5Z
6zsGDB9GqVSv94xUrViA2Nha3b98W/XwZGRmoVasW1q9fj0gTf5nz8vKKJE+5ubkICQlhnyMiF6RbrQYY3yjWlgM
G2devY1jv3vhl3z4cW7oUdfv0kW77i9BQ7ZI7Y//7d8aePdzywyUptkP28OHD0adPn1LPCQ0Nlez5AgMDUatWLZw
tZamqWq2GWq2W7DmJSLl0Aw/GGixLsbVGaZ/Z/gEB2JiUhAsXLqBu3bpleyJDuummnj21iVDxrE8QgB49tIE5SxK
hq1sispCsyZG/vz/8rVmSaqUbN24gLS0NgVxuSUQWiowEIiJss82ZuV0tVCqVtImRjqmsr1w5bcYWH6/9Usq+YYZ
ZZvXq2mOZmRwlIqspZvuQy5cv4+bNm/jxxx8xe/Zs7Hq0kqJevXr6ZmgNGzbEjBkz0LxcXHo0aMHAgM
DcfHiRUycOBGXL1/G6dOnUalSJYuel9uHEJHUTO1qASQDaISEhAD75CO6pOKHH7TJUHFSzh9KPbVlGPuaNUBWlvH
zlJLgkeTK9PktKMSAAQMEaBduFPlKSkrSnwNAWL58uSAIgnDv3j2hU6dOQrVq1QR3d3fhsccEwYMGCBcvnxZ1PP
m5OQIAIScnBwJXw0RuaqHDwUhOFgQtKmR4dcVAagiAP6Cn1+C8PCh7AFpv1QqQQgJEcoUUEJCyecIDtYel+p+pcW
vUln/XKRYZfn8VszIkVw4ckREUjK+p6oAoAuAbY8e90FS0jr7lMnYekNaqTZ/NXe/0jhjkTmZxY1niYgUwnj7oEX
4NzEKArDAftt02bLbpdR7sJR2v9JwY1gSickREZEdlVwPchbAWIPHywD42m+bLlv2PZJ6DxZz9zOHG8Oap9FoRxP
XrdN+l3TzQOVgckREZEdFm0s+hLYP9r1Hvx0GlaqzfbfpCgsD/PxKP8fPz7qApB6VKmtyw5XKpUtM1PbACg8H+vX
Tfg8NlWYjYoVhckREZEdF93+dDWDvo5/rQreTmtNs0yX1qJS1yQ03hjVPV8tVfGQuPV173MUSJCZHRER2FhkJzJ6
dAmDKoyPlAKxCSEgF+2/TtWsXcONG6efcuGFdvY7Ue7CYu5+p5wCcKOO0Aalrw5wAkyMiIjsrKCjAihVRAAoAAK+
9Nh5JSe3k2abLllNaUm/+Wtr9TOHGsOZJXRvmBJgcERHZmbu7Oz744ANUrVoVDB5kGNmw9pWWLzV+
N3a9aNe3oxv/+p/3ixrCWs1fNl4Kwz5EZ7HNERLZy9epV3LlzBw0aNJNJAvCHttRGurDtlybCbrbBvZ2rrXlUzK8vn
N5MgMJkdE5PR0xbhAyY1oAU5LGbJkUzylsVeCbGdsAklEpACHDx+WOwTjpJ76clbOuqJL6towJ8DkiIjIDjZv3ox
WrVrhP//5D27duiV3OCVFRgIXL2qnTlivU5Kzr+higlwEp9XM4LQaEZVVdnY2GjdujOvXrwMAli9fjujoaHTo
ctC6nBCeqpyrL53d5G8VEREQABEHAsGHD9InRyy+/jEDBsgclYI4yoe1q6zocnNTdnInESZHREQ2tG7dOnz33Xc
AAF9fX3z99ddddQiWli6MocqfjZ1i0PyKFwWs0MTqsRkbWuXLmCJk2a4PbtwCAjRs3olevXvDy8sKECRPQ
jRsDf334jSk66osuZcVqNiMjBCIKAt956S58Y9evXj4mRpSwpfu7Tp2jxs61HlHQrunr21CZCxloeuNiKLmfG1Wp
ERDbw5ZdfYvv27QoAoKAgfPHFFzHpCDmtrMASq4Ks8dyeq7ochmcVjOD02pEJNbff/+N5s2b4/79+wCArVu3
nTjHpSDr1gH9+om/zl5TW45SJE6lYhNIIiIHUqlSJYQ/Wvb9zjvvMDESy9qiZnttkKpb0dW3L+TbFI9siTVHREQ
SCwwMxM8//4w4w1a9age/fucoejPGFh2hEgU8XP5ih9OT3JjiNHREQ2oFKp8J///Ac/cVKlSxkW07C0twOT2VEZM
jIiIJPHjwADdv3pQ7DOdhqvi5tCkslQoICdGOPBGVAZMjIiIJTJ48GU2aNMHDhw8BAFu3bjtxjpSDr1gH9+om
2W5B167TflbovGkmONUdERGW0a9cufprppxAEAREREUhNTUUgp3akYWw7Czc3452z4+MtX07vSN23yeEwOSIiKoM
7d+5gwIAB0HVFmTZtGhMjW4uMBCIirF9Ob6r7tq5XEnsWuTz2OTKDfY6IqDRDhgzBkiVLMGDAAMyYMQMAcPr0
```

Ltw2IqSaT3AbEabDPERGRDDZv3qxPjCpUqICVK1cyMbKGPWt/zHXftlevJHJonFYjIrJCdnY2Bg0apH88d+5c1Kl
TR8aIFMretT+W9kBirySXxpEjIiKRBEHAsGHDcP36dQDAyy+/jLffflvmqBRIV/tTfCTHlvukWVoPxroxl8aaIzN
Yc0RExa1duxZvvvkmAMDX1xcnT55kEbZYctX+6J7XVPdt1hw5DdYcERHZUWFhISpWrAgAWLRoERMja9i69sdUHVN
p3bet6ZVETonJERGRSP/5z39w/PhxzJgxA7169ZI7HGWyZe1PYqJ2dCg8HOjXT/s9PTPTfaeQ+4wnIctan/Yw4gMsOa
IiMiGHj58iPDwcAwdOhR3796VOxznYIvaH/YwIokwOSIiMuOTTz7Bvn37sBHjxYtSFWqZ7Jr2tYYSma2p/SGkai2p/SGkayhxFJhE0giYh
KceTIEUydOhUAUK5cOUyZMkXmiOzM1nU5YvZJM9cwkj2MSCJMjoiITHjw4AGioqLw8OFDAMCECRPQpk0bmaOyI3sImKx9kzRbQmdD1RqTnw4Yczcgs8//wQAtGjRApMnT5Y5IjuSo3u1KRqNNkkzlvDojsX
Gar+zhxFJgMkREZERO3fuxJw5cwAAarUaq1evhoeHh8xR2YmlyGtN4g1JKbQmj2MSAKcViMiKiY3NxfR0dHQdTr
5+OOP8eSTT8oclR2JSUbE9CCylthCazF1TERGMDkiIipm9Ojxj9IRuHjxIgDg+eefx6hRo+QNyc4cbdWXNYXWXlXltQxEZn
AaTUiIgN5eXk4f/48AKBixpYsWIF3wslizEbMHRVn2FhhWmVnYzFFhrzXEemoVEBICAutSbsMaP7Fjxw58/vnnn
lCmrXri13SPanS0YcZdWXrmFkz57a5zaMiYXWZAMcOSIqZcuXKIjY1FdHS03KHKHIwxF3rmehNdkR91Yzg3urEbk
GQRCgMjVt46qM9TkKCdEmRnIlI9wslizEjWdtiMkRkfNLS0tDREQE4uPj8sdjmNhMkIK5fQbz168eBGDBg1
C7dq14eXlhcp162LKlCnIz88H+2LKlCnIz88v9TpBAB69evlzs
kx6Jb9dW3r/Y7EyNyAYpIjv766y8UFhbiq6++wlTpzB37lwEDWmEdygtLvvKdfTMyU5tXb5yRfUcpZtqclsHkxwk2GjSBvpnjCenjBrA7rqNKYkEvBAxk6UKQXuFBhDDmjkGy6xqcmNOUHsfTjq5CcE7gB3RMFYWvBD+9Wmbm5iRKjfh73D8Xl91k1dNfjiiyJRNC27Z51
RDQgIwMyZMzF06FCLnofTakTO68yZM2jRogUHnDNWARSBoEsxSRoNotPHPl31ozmazXD7N59G8Ys5MoXpVuaaQyyYdkhKQ5tBw4+ZYGEEKVt8LRE94TTBy
7dg2dOnXSH1Or1Wjfvj327Nlj8rq8vTk5uYW+SIi5/Pw4UP0799fnxgNHz7ceRKjxEQgNBQIDwf69dN+Dw1135ZNmV3UdqqOTIqafzz4+9U76
oRAqmyOTo/PnzmD9/PmJiYkyec+3aNBAQEBAkeMBAQH63xkzY8YM+Pj46NKoHFpdbZNRzY1K2W9aIp7ovSdlf4jOCvYXgcE9/PF8D7SQrxo9rrpd7POoIcgq
PmTNnyhyRRBxpw1gihZI1OYqLi4NKpSr169YWJJKA2bfXUTFnNzpfhDmoS2ZWQnMZjv5ZY2G2gzK0Mq4ft47Q2Aam
SrHtxROSwDh8+jKlTpwLQ9jRauXIlvL29TfPh8CJX3p21YLU/66aiiw8ZxY2xS9LeQc8NIWKfPJ4Kt8A
8uNTratSoAUA7ghRo0OI+MzOzxGiSIbVaDeqxlsFVB+ZS/VA4Rc8y5a0QxvfXzPIzPbQ2xL1r1ZGpSRQsnaIXv48OHo06dPqeHhcbqf7569SrCw8PRtm1F
8uNTratSoAUA7ghRo0OI+MzOzxGiSIbVaDbVabUH0RKRE9+/fR1RUFB4+fNwgLdL0fwkUTa0
I398f/v7+Fp2bnp6O8PBwtNWxuqrBk66o45swMEjDjZbC2qbvffbjkGczSZXQlnCwA9k4Pd5Fbb8FDeLOk+Fp6ptVzrDhG4dRPVSoFfJnxtXeKpyEy2+yLHYD8ZKRPQYFK25w
2Dp7BMTKtrg2tT4mTS8yRHtGDMpH2PPyYSQT9lyMC0Rmx4ZKbk68zPsMPAw2ecIwL21D5zx6Mf5p7Sab8jLkqwYC4Go1v+ORe7l
PHjJHRYqUmKjdksWw83hwMDBvHovbnRin1YjI6SxYsAA7duwAAAQHB+Pzzz+XOSJSJG7i67KYHBGRU/nrr7/w3nv
v6R8vX74EVapUcWUVhURkVOJj4/X9zP673//ixdffFHmiEiRxGziS06HNUdE5Ff++OOSOLBAcHY8OGDfj
kk0/kDoeUipv4uhWOHBGR0wsJCcGBAwf0i3H2tTtVsNQYMAugAOHrFLAgMLOCmrIUYc1WGFcEdIQr4bbHdnPuG
+ToyrJEREp3pw5c3Dw4EG5uamXa4PlEyQdI/j49nvyEkxOSIiRUtOTYb64+9SwPRtm1bzJgxQ+5WyJlwE1+XxJ
sIlKs3NxcREsREdHQxAEAAEsEO2i2FyZIbwqJ9Fbm6uzJEQUXHvvvvbbb/GvvsuL26BAB49tln8dZbb/G
/VbKNp5769+d//pEvDrKY7v8FgrFeVWaoBGuuciEXLlxA3bp15Q6DOTLD19caD3bp15Q6DiIiIrHD+/HnUqVqL1/vvvyx2WouX
4+PjIHI2y5ebmIiQkBGlpaahCubLc4SgW30fp8L2ugL2UDt9LafB9OTg4eGEyOzChXTluz7uPjw7+oEq
cuTLfSwnwfZQO30p8L2UBt9H6eg+x0VdY4M4iIiIiBSLyRERERGRCAEFhRERGRoa2qQRkGGtyZZMoVLhCXA91IafB+lw/dSOnw
vpcH3UTpleS+5Wo2IiIiIiIyIiIyIywOSIiIiIyIiIyACTIyIiIDTI5ECA0NhUqlKvKvvL1/vvvyx2WouX
l5aF58+ZQqVRISUkpcTg48wU9YrIOE5ZOMk1kDeYlWM5NlZlYMl1VYGYVLiWSVR+sWIDUVm0iRpMZ+Z02...

4gAMHDuC5555D1apVceHCBUyePBl169Z1+VEjMa5evYoOHTrgsccew6effoqsrCz972rUqCFjZMp0+fJl3Lx5E5c
vX4ZGo9H3MKtXr57+v3cqafTo0YiKikKrVq3Qtm1bLF68GJcvX2btm0h3797FuXPn9I9TU1ORkpICX19fyz+vpVw
y58wOHz4stG7dWvDx8RE8PT2FBg0aCFOmTBH++ecfuUNTvNTUVC7lt9Lx48eF8PBwwdfXV1Cr1UJoaKgQExMjXLl
yRe7QFGX58uUCAKNfJN6AAQOMvpdJSUlyh+bwFixYINSqVUvw8PAQnnrqKWHnzp1yh6Q4SUlJRv/+2RgwwOJ7uHz
NEREREhTqoTERERGWByRERERGSAyRERERGRASZHRERERAaYHBEREREZYHJEREREZIDJEREREZEBJkdEREREBpg
cEZFZ0dHRojdutFRycjJUKhU3J5XAihUrUKVKFbnDIFI8JkdERERBpgcEVGZzZkzB02aNEGFChUQEhKCd955B3f
v3tX//tKlS+jWrRuqVq2KChUq4Mknn8SWLVtw8eJHFhIeHAwCqVq0KlUqF6Ohok8+ze/dutG/fHt7e3qhatSo6d+6
MW7duAQDy8vIwYs8vIwYsQIVK9eHZ6ennjuuedw8OBB/bW6EaqtW7eiRYsw8PLyQseOHZGmYlffvkFjRo1QuXKldG3b1/
cu3dPf12HDh0wfPhwDB8+HFWqVIGfnx8+/PBDGO68dOvWbcXV/Pv3x9p8x9Lq1aFt7c3unbtirNnz+p/rxvR2bp1Kxo1aoS
KFSuiS5cuyMjIKPL6li9fjkzEv5Ll8uRDz9vNfVrRw9V+EDb3fgCGbEwoUL9b+7ePEiVCoVEhMTER4eDm9vbzRr1gx79+7Vv76BAwciJyc
HKpUKKpUKcXFxIv4UiJjPRvu+EZETGTBggARREWHy93PnzhV+CvKFC8KOOHTuEBg0aCMOGDdP//u+++07o8/8VFVFbnDIFI8JkdERERBpgc
l4fjx48L58+eFn376Sdi5c6c6fw8OFDISEhQQAgnDlsRsjIyBBu375t9DmOHj0qqNVq4QL4xqqNVqYdiwYUJKSopw8uRJJKSopw8uRJYf78+UJ
WVpYgCIIwYsQIIIwSgoSNiyZYtw6tQpYcAULVqlWFGzduCILw72aOKLw72aUbdq0Ef744w/744w/hyJEjEjQr169YT27dsLnTp1Eo4
cOSL8/vvvgp+fn/DJ5jBe+ZNAOn7d9+/ZCxYoVhEwpRwp//fWX8M033jeD3tprDQGNjYTff/9dSElJETp
37izUq1dPyM/PFwRBu7Gtu7u78OKLLwqDhx8LDRq8IdvRq1Ejo16+f/h6LFy8Lzg4WAhtZHhYOSHQ4QhtCQkKC4OvrK6x
YsUIQhH83aG7YsKHw888/PF69uwuPbnBqeBuU9jRQnqa+/vpr/e9NQ8PWzfGVIiwmnPOeExudIVmH/0Zh8wlDrhB+GNc7sB9cHv+uwi+I2CW
is8wlR8Vt3LhR8PPz0z9u9nqSJEBcXZ/RcXdjJy69atUu/Zt29f4dl4dnnzX6u7t37wru7u7u7CmjVr9Mfy8/8OFoKAgYda
sWUWe53//+nBkzZggAhPPz+uPDR06+vPuPd7jPBDR06dPzZ9998CAGH37t3
632dnZwteXl7Cxo0bBUHQJkcAhHPnzunP+i1JJjCPnzjmIyXhYHm9rlPCXjBBVhQQJkcAhMPnzunPWbDggRAQEKB/HBISQ31YTJEvpr9PPnTo
lABBOnz6tfx4fHx+j7xERWY7TakRUZklJSXjppddQs2ZNVKpUCf/3798eNGzfg1JtNPAMUztKHqLmqRqK
C48ePi36Ol1JQUvPDCC0Z/d78eRQUFZZ5/VH3N8d8czzzyD0Om3adOm+p8L+5wK4wtCgRX/Qqw7ffq8yQXQPN+MA/EHAgOPE7ANBNOnXqFDmWmWmWmZlZnpc
2bdpApVLpH7dt2xZPwt+rEQC8udK3sVZ+vzB/f2/6ZdkfEkozNZK/XcBgnLtEnqRjYGpJlxJVpGXbWYjrbWFqbrcpqnKk1r2fKGdTk+bk9iR
KFSvqvz7++GOcP3/pPzOUy0yyG7NfmUV/+P9/NieGnF3W+eMeLoZvbMP+AbLxlwSvFqpNh5rQLAwxBiXjWGd8pFBSOc+vQAjYv8+HDHmJDzm+Y
AVFQUTpw4gVatWmH+/PminsfLy9+p9VKlWRx7pjhYWFFsclGNQelfbcxp5Hd63u+ZY
sWYKUlBT918mTJ7Fv375S8noikweSIiMrk0KFDePjwIaRQSrDjgYYGQmLMmDF
YsmQJAMDDwwMAoNFoSn28ueOHUZ/V69ePxh4eOCPP/7QHQCPP7QHysoKMChQ4fQQelFEja1+aXvEEZd++fXj6u
GJ554Ag8fPsT+/x4wb+/vcvi587ICAANWvxYv9v8ZiCAA6NWvmpCxlC8wvJLrb+3o1FixCXVOGlaraVpCpoF93zBF37Y3b5aJbvOU
SklLkmK+vL+rWrYuHDx9i/vz56NatG3bv3FixYVOS82NhZdu3ZF/fr1/vz9/H6xKFWrVpQmVT4+eef8L
LL8PLywsVK1ZYKwSCpKIH52KlLmaW9FnM6I8F6o6vvn7Ql+6qd4m9ZIZb2V6Nvj2HHhmHchHHw9fXFY489hlmzZuZHevVXsYNGh
QmV97WloaRo8ejaFDh+LIkSOkSOYO8QRK38+hK2Pc8VvvcSMwDA448/joiICAwePBhfffffffUVKlWqhPfffx81RERDSWBKc8hRFxeH+Edaeh8tjd4Vz4Mzl2BiFqUqjXX8+meNfhfzXxk+b8+rN3+6oJ0GBBWnlkVvk0aeqNO88Dl36dKA2OCo7RnODmIiyN2wgOrCmHlTqvTbHyy1KMIRORSF2CXEJ07Zl6VkWfaQ4JamvFExEpwoABBAwQAJb4
GDBggCIIgzJkzRwgMDBS8vLyEzp07C6tWrSSpSZD18+HChbt26glqtFqpVqyZERUJ2dnZ+vt/9NFHQo0aNQSVSqW
/pzHJyclCu3btBLVaSpUkXo3Lmz/jnu378v/Pe//xX0FtVotPPvss8kKBAwf01xor/DZWwDxlyhShWbNm+sf
t27cX3nnnnHSEmJkaoXLmyVqVeH9998vUqB98+ZUqB98+ZNISoqLwAA7V7uLRyAAQLrmCRmVLYx2dvvTQhDx97P9eX5BqlJ/cSy4qhAWtS
QqlatKjz//PNCYmKiAj/FmQfPXpUf/6tW7EAEJSUpL+WEnGzXpCSCYGLCnIiI0KFDBzR
v3hzx8fFyh0JEiI0kPFDsSKaIyIiIDTIiIiIDHBaiIiIDZLQSIiIjA5IiIiIMLQyIiIQJiMiIiIgNMjoiIiIMMgiMgA0yOiIiIj8P6qdWLO8yOlJAAAAAElFTkSuQmCC",
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
]
},
"metadata": {},
"output_type": "display_data"
}
],
"source": [
"plot_scatter(test[1], test[2])\n",
"# title('Test data')\n",
"plt.plot(x,y, linestyle='--', linewidth=2, color='k')\n",
"plt.xlim(-5, 1)\n",
"plt.ylim(-2.2, 1)"
]
},
{

```
"cell_type": "code",
"execution_count": 41,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Confusion matrix:\n",
"[[46  5]\n",
" [ 4 45]]\n",
"\n",
"Accuracy: 0.910\n"
]
}
],
"source": [
"# Print confusion matrix\n",
"conf = np.array([\n",
"    [(apply_lda(test[1], W, b) == 1).sum(), (apply_lda(test[2], W, b) == 1).sum()],\n",
"    [(apply_lda(test[1], W, b) == 2).sum(), (apply_lda(test[2], W, b) == 2).sum()],\n",
"])\n",
"\n",
"print('Confusion matrix:')\n",
"print(conf)\n",
"print()\n",
"print('Accuracy: %.3f' % (np.sum(np.diag(conf)) / float(np.sum(conf))))"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.16"
},
"orig_nbformat": 4
},
"nbformat": 4,
"nbformat_minor": 2
}
```

File: eeg_process.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"import numpy as np\n",
"import matplotlib.pyplot as plt\n",
"import scipy.io\n",
"%matplotlib inline"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"m = scipy.io.loadmat('CLASubjectA1601083StLRHand.mat', struct_as_record=True)\n",
"print(m.keys())"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"EEG = m['o']['data'][0][0].T\n",
"list_markers = [s[0] for s in m['o']['marker'][0][0]]\n",
"labels = np.asarray(list_markers)\n",
"\n",
"sample_rate = m['o']['sampFreq'][0][0][0][0]\n",
"# Loading channel names\n",
"channel_names = [s[0][0] for s in m['o']['chnames'][0][0]]\n",
"nchannels, nsamples = EEG.shape\n",
"list_markers = [s[0] for s in m['o']['marker'][0][0]]\n",
"unique_events = np.unique(list_markers)\n",
"\n",
"print(\"EEG Dimensions: \", EEG.shape)\n",
"print(\"Label Dimensions: \", labels.shape)\n",
"print(\"Sample Rate:\", sample_rate)\n",
"print(\"Number of channels:\", nchannels)\n",
"print(\"Channel names:\", channel_names)\n",
"print(\"Event codes:\", unique_events)\n"
]
},
{
```

```
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"from matplotlib.collections import LineCollection\n",
"\n",
"def plot_eeg(EEG, vspace=200, color='k'):\n",
"    '''\n",
"    Plot the EEG data, stacking the channels horizontally on top of each other.\n",
"\n",
"    Parameters\n",
"    ----------\n",
"    EEG : array (channels x samples)\n",
"        The EEG data\n",
"    vspace : float (default 100)\n",
"        Amount of vertical space to put between the channels\n",
"    color : string (default 'k')\n",
"        Color to draw the EEG in\n",
"    '''\n",
"    \n",
"    bases = vspace * np.arange(22)\n",
"    \n",
"    EEG = EEG.T + bases\n",
"    \n",
"    # Calculate a timeline in seconds, knowing that the sample rate of the EEG recorder
was 200 Hz.\n",
"    samplerate = 200.\n",
"    time = np.arange(EEG.shape[0]) / samplerate\n",
"    \n",
"    # Plot EEG versus time\n",
"    plt.plot(time, EEG, color=color)\n",
"\n",
"    # Add gridlines to the plot\n",
"    plt.grid()\n",
"    \n",
"    # Label the axes\n",
"    plt.xlabel('Time (s)')\n",
"    plt.ylabel('Channels')\n",
"    \n",
"     # The y-ticks are set to the locations of the electrodes. The international 10-20
system defines\n",
"    # default names for them.\n",
"    plt.gca().yaxis.set_ticks(bases)\n",
"    plt.gca().yaxis.set_ticklabels(channel_names)\n",
"    \n",
"    # Put a nice title on top of the plot\n",
"    plt.title('EEG data')\n",
"\n",
"\n",
"# Testing our function\n",
"plt.figure(figsize=(25, 12))\n",
"plot_eeg(EEG, 300)"
]
```

```
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"labels.shape"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"non_zero_i = np.flatnonzero(labels)\n",
"non_zero_i"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"onsets = [non_zero_i[0]]\n",
"for i in range(1, len(non_zero_i)):\n",
"    if non_zero_i[i - 1] != non_zero_i[i] - 1:\n",
"        onsets.append(non_zero_i[i])\n",
"onsets = np.asarray(onsets)\n",
"onsets"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"plt.figure(figsize=(300, 100))\n",
"plot_eeg(EEG, 400)\n",
"for onset in onsets:\n",
"    plt.axvline(onset / 200., color='r')\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"# Creating time slices with a window of 1 sec\n",
"time_slices = [(s, int(s + 1* 200)) for s in onsets]\n",
```

```
"print(\"Length of time_slices:\", len(time_slices))\n",
"print(\"First 10 trials are:\")\n",
"time_slices[:10] # Showing 10 trial onsets and ending times"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"trials = [EEG[:, s:e] for s, e in time_slices]\n",
"trials = np.asarray(trials)\n",
"print(\"Shape of trials:\", trials.shape)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"plot_eeg(trials[0], 20)\n",
"print(trials[0].shape)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"from scipy.signal import stft\n",
"\n",
"def apply_stft(signal, sampling_rate, window_size=64, overlap=0.5):\n",
"    \"\"\"\n",
"    Apply Short-Time Fourier Transform (STFT) to a given signal.\n",
"\n",
"    Parameters\n",
"    ----------\n",
"    signal : array\n",
"        Input signal\n",
"    sampling_rate : int\n",
"        Sampling rate of the signal\n",
"    window_size : int (default: 64)\n",
"        Size of the STFT window\n",
"    overlap : float (default: 0.5)\n",
"        Overlap between consecutive STFT windows\n",
"\n",
"    Returns\n",
"    -------\n",
"    stft_output : array\n",
"        STFT output\n",
"    \"\"\"\n",
```

```
"    overlap = int(window_size * overlap)\n",
"            f,   t,   stft_output   =   stft(signal,   fs=sampling_rate,   window='hann',
nperseg=window_size, noverlap=overlap)\n",
"    return f, t, stft_output\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"channel_names[4]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"# Apply STFT to the first trial's C3 channel\n",
"f, t, stft_output = apply_stft(trials[0][4, :], sampling_rate=200, overlap=0.6)\n",
"\n",
"# Plot the spectrogram\n",
"plt.pcolormesh(t, f, np.abs(stft_output), cmap='plasma')\n",
"plt.ylabel('Frequency [Hz]')\n",
"plt.xlabel('Time [sec]')\n",
"plt.title('Spectrogram of the first trial\\'s C3 channel')\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"for trial in trials:\n",
"    print(trial.shape)\n",
"    break"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"def avg_stft(trials, channels, sampling_rate, window_size=64, overlap=0.5):\n",
"    \"\"\"\n",
"    Compute the average STFT across selected channels for each trial.\n",
"\n",
"    Parameters\n",
"    ----------\n",
```

```
"    trials : array\n",
"        EEG data trials\n",
"    channels : list\n",
"        List of channel indices to compute the average STFT for\n",
"    sampling_rate : int\n",
"        Sampling rate of the EEG data\n",
"    window_size : int (default: 64)\n",
"        Size of the STFT window\n",
"    overlap : float (default: 0.5)\n",
"        Overlap between consecutive STFT windows\n",
"\n",
"    Returns\n",
"    -------\n",
"    f : array\n",
"        Array of frequencies\n",
"    t : array\n",
"        Array of time points\n",
"    avg_stft : array\n",
"        Array of averaged STFT values across selected channels\n",
"    \"\"\"\n",
"    stft_all = []\n",
"    for trial in trials:\n",
"        stft_trial = []\n",
"        for channel in channels:\n",
"                                f,   t,   stft_output   =   apply_stft(trial[channel,   :],
sampling_rate=sampling_rate, window_size=window_size, overlap=overlap)\n",
"            stft_trial.append(np.abs(stft_output))\n",
"        stft_trial = np.asarray(stft_trial)\n",
"        stft_avg = np.mean(stft_trial, axis=0)\n",
"        stft_all.append(stft_avg)\n",
"    stft_all = np.asarray(stft_all)\n",
"    # avg_stft = np.mean(stft_all, axis=0)\n",
"    return f, t, stft_all\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"channels = [3, 4, 5, 6, 18, 19]\n",
"f, t, stft_all = avg_stft(trials, channels, sampling_rate=200, overlap=0.6)\n",
"\n",
"plt.pcolormesh(t, f, abs(stft_all[0]), cmap='plasma')\n",
"plt.ylabel('Frequency [Hz]')\n",
"plt.xlabel('Time [sec]')\n",
"plt.title('Average spectrogram for selected channels')\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
```

```
"outputs": [],
"source": [
"trials[0][4, :].shape"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"for i in range(5):\n",
"    plt.pcolormesh(t, f, stft_all[i], cmap='plasma')\n",
"    plt.ylabel('Frequency [Hz]')\n",
"    plt.xlabel('Time [sec]')\n",
"    plt.title('Average spectrogram for selected channels')\n",
"    plt.show()\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"stft_all[0].shape"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"non_zero_labels = [labels[onset] for onset in onsets]\n",
"print(\"Number of trials:\", len(non_zero_labels))"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"print(\"stft_all shape:\", stft_all.shape)\n",
"print(\"labels:\", len(non_zero_labels))"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
```

```
"import os\n",
"import matplotlib.pyplot as plt\n",
"\n",
"# Create a dictionary to map labels to class directories\n",
"label_to_class = {1: '1', 2: '2', 3: '3'}\n",
"\n",
"# Create the class directories if they don't exist\n",
"for class_dir in label_to_class.values():\n",
"    if not os.path.exists(class_dir):\n",
"        os.makedirs(class_dir)\n",
"\n",
"# Iterate over all trials and generate STFT plots\n",
"for i, trial in enumerate(stft_all):\n",
"    # Generate the STFT plot\n",
"    plt.pcolormesh(t, f, abs(trial), cmap='plasma')\n",
"    plt.ylabel('Frequency [Hz]')\n",
"    plt.xlabel('Time [sec]')\n",
"    plt.title(f'Trial {i+1} - Average spectrogram for selected channels')\n",
"    # Get the corresponding label for the trial\n",
"    label = non_zero_labels[i]\n",
"    # Get the class directory for the label\n",
"    class_dir = label_to_class[label]\n",
"     # Save the STFT plot as an image with the label in the filename in the respective
class directory\n",
"    plt.savefig(os.path.join(class_dir, f'trial_{i}_label_{label}_stft.png'))\n",
"    plt.close()\n"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Model: \"sequential_1\"\n",
"_________________________________________________________________\n",
"Layer (type)                 Output Shape              Param #   \n",
"=================================================================\n",
"conv2d_3 (Conv2D)            (None, 222, 222, 32)      896       \n",
"_________________________________________________________________\n",
"max_pooling2d_3 (MaxPooling2 (None, 111, 111, 32)      0         \n",
"_________________________________________________________________\n",
"conv2d_4 (Conv2D)            (None, 109, 109, 64)      18496     \n",
"_________________________________________________________________\n",
"max_pooling2d_4 (MaxPooling2 (None, 54, 54, 64)        0         \n",
"_________________________________________________________________\n",
"conv2d_5 (Conv2D)            (None, 52, 52, 128)       73856     \n",
"_________________________________________________________________\n",
"max_pooling2d_5 (MaxPooling2 (None, 26, 26, 128)       0         \n",
"_________________________________________________________________\n",
"flatten_1 (Flatten)          (None, 86528)             0         \n",
```

```
"_________________________________________________________________\n",
"dense_2 (Dense)              (None, 64)                5537856   \n",
"_________________________________________________________________\n",
"dense_3 (Dense)              (None, 1)                 65        \n",
"=================================================================\n",
"Total params: 5,631,169\n",
"Trainable params: 5,631,169\n",
"Non-trainable params: 0\n",
"_________________________________________________________________\n"
]
}
],
"source": [
"from keras.models import Sequential\n",
"from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense\n",
"\n",
"# Define the input shape of the CNN\n",
"input_shape = (224, 224, 3)\n",
"\n",
"# Define the CNN architecture\n",
"model = Sequential()\n",
"model.add(Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))\n",
"model.add(MaxPooling2D((2, 2)))\n",
"model.add(Conv2D(64, (3, 3), activation='relu'))\n",
"model.add(MaxPooling2D((2, 2)))\n",
"model.add(Conv2D(128, (3, 3), activation='relu'))\n",
"model.add(MaxPooling2D((2, 2)))\n",
"model.add(Flatten())\n",
"model.add(Dense(64, activation='relu'))\n",
"model.add(Dense(1, activation='sigmoid'))\n",
"\n",
"# Compile the model\n",
"model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])\n",
"\n",
"# Print the model summary\n",
"model.summary()\n"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Found 851 images belonging to 3 classes.\n",
"Found 109 images belonging to 3 classes.\n"
]
}
],
"source": [
"import cv2\n",
```

```
"from keras.preprocessing.image import ImageDataGenerator\n",
"\n",
"# Define the path to the directory containing the STFT plot images\n",
"train_dir = 'train'\n",
"test_dir = 'validation'\n",
"\n",
"# Define the image size and batch size\n",
"img_size = (224, 224)\n",
"batch_size = 32\n",
"\n",
"# Define the ImageDataGenerator object for preprocessing the images\n",
"datagen = ImageDataGenerator(\n",
"    rescale=1./255,\n",
"    preprocessing_function=lambda img: cv2.resize(img, img_size))\n",
"\n",
"# Define the training and validation generators\n",
"train_generator = datagen.flow_from_directory(\n",
"    directory=train_dir,\n",
"    target_size=img_size,\n",
"    batch_size=batch_size,\n",
"    class_mode='binary')\n",
"\n",
"valid_generator = datagen.flow_from_directory(\n",
"    directory=test_dir,\n",
"    target_size=img_size,\n",
"    batch_size=batch_size,\n",
"    class_mode='binary')\n"
]
},
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"data batch shape: (32, 224, 224, 3)\n",
"labels batch shape: (32,)\n"
]
}
],
"source": [
"for data_batch, labels_batch in train_generator:\n",
"    print('data batch shape:', data_batch.shape)\n",
"    print('labels batch shape:', labels_batch.shape)\n",
"    break"
]
},
{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
```

```
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Epoch 1/30\n",
"27/27 [==============================] - 5s 189ms/step - loss: 0.2959 - accuracy:
0.3608 - val_loss: -1.2969 - val_accuracy: 0.3028\n",
"Epoch 2/30\n",
"27/27 [==============================] - 5s 186ms/step - loss: -0.0973 - accuracy:
0.3608 - val_loss: -3.7730 - val_accuracy: 0.3028\n",
"Epoch 3/30\n",
"27/27 [==============================] - 5s 186ms/step - loss: 0.0857 - accuracy:
0.3643 - val_loss: -3.2833 - val_accuracy: 0.3028\n",
"Epoch 4/30\n",
"27/27 [==============================] - 5s 185ms/step - loss: -2.8298 - accuracy:
0.3608 - val_loss: -17.4720 - val_accuracy: 0.3028\n",
"Epoch 5/30\n",
"27/27 [==============================] - 5s 188ms/step - loss: -17.8126 - accuracy:
0.3737 - val_loss: -121.3252 - val_accuracy: 0.3028\n",
"Epoch 6/30\n",
"27/27 [==============================] - 5s 187ms/step - loss: -194.7553 - accuracy:
0.3608 - val_loss: -1026.3634 - val_accuracy: 0.3028\n",
"Epoch 7/30\n",
"27/27 [==============================] - 5s 190ms/step - loss: -914.5147 - accuracy:
0.3690 - val_loss: -6997.7612 - val_accuracy: 0.3028\n",
"Epoch 8/30\n",
"27/27 [==============================] - 5s 187ms/step - loss: -6816.2866 - accuracy:
0.3608 - val_loss: -39421.5195 - val_accuracy: 0.3028\n",
"Epoch 9/30\n",
" 6/27 [=====>.....................] - ETA: 3s - loss: -36509.7383 - accuracy:
0.3575"
]
},
{
"ename": "KeyboardInterrupt",
"evalue": "",
"output_type": "error",
"traceback": [
"\u001b[1;31m---------------------------------------------------------------------------
\u001b[0m",
"\u001b[1;31mKeyboardInterrupt\u001b[0m                                Traceback (most recent
call last)",
"Cell \u001b[1;32mIn[10], line 1\u001b[0m\n\u001b[1;32m----> 1\u001b[0m history
\u001b[39m=\u001b[39m model\u001b[39m.\u001b[39;49mfit(train_generator,
epochs\u001b[39m=\u001b[39;49m\u001b[39m30\u001b[39;49m,
validation_data\u001b[39m=\u001b[39;49mvalid_generator,
verbose\u001b[39m=\u001b[39;49m\u001b[39m1\u001b[39;49m)\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\keras\\engine\\tr
aining.py:1184\u001b[0m, in \u001b[0;36mModel.fit\u001b[1;34m(self, x, y, batch_size,
epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,
validation_freq, max_queue_size, workers, use_multiprocessing)\u001b[0m\n\u001b[0;32m
```

```
1177\u001b[0m                                                    \u001b[39mwith\u001b[39;00m
tf\u001b[39m.\u001b[39mprofiler\u001b[39m.\u001b[39mexperimental\u001b[39m.\u001b[39mTra
ce(\n\u001b[0;32m                              1178\u001b[0m
\u001b[39m'\u001b[39m\u001b[39mtrain\u001b[39m\u001b[39m'\u001b[39m,\n\u001b[0;32m
1179\u001b[0m     epoch_num\u001b[39m=\u001b[39mepoch,\n\u001b[0;32m   1180\u001b[0m
step_num\u001b[39m=\u001b[39mstep,\n\u001b[0;32m            1181\u001b[0m
batch_size\u001b[39m=\u001b[39mbatch_size,\n\u001b[0;32m         1182\u001b[0m
_r\u001b[39m=\u001b[39m\u001b[39m1\u001b[39m):\n\u001b[0;32m          1183\u001b[0m
callbacks\u001b[39m.\u001b[39mon_train_batch_begin(step)\n\u001b[1;32m-> 1184\u001b[0m
tmp_logs                                                    \u001b[39m=\u001b[39m
\u001b[39mself\u001b[39;49m\u001b[39m.\u001b[39;49mtrain_function(iterator)\n\u001b[0;32
m                          1185\u001b[0m                            \u001b[39mif\u001b[39;00m
data_handler\u001b[39m.\u001b[39mshould_sync:\n\u001b[0;32m       1186\u001b[0m
context\u001b[39m.\u001b[39masync_wait()\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\pytho
n\\eager\\def_function.py:885\u001b[0m,                                                in
\u001b[0;36mFunction.__call__\u001b[1;34m(self, *args, **kwds)\u001b[0m\n\u001b[0;32m
882\u001b[0m                          compiler                          \u001b[39m=\u001b[39m
\u001b[39m\"\u001b[39m\u001b[39mxla\u001b[39m\u001b[39m\"\u001b[39m
\u001b[39mif\u001b[39;00m       \u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_jit_compile
\u001b[39melse\u001b[39;00m
\u001b[39m\"\u001b[39m\u001b[39mnonXla\u001b[39m\u001b[39m\"\u001b[39m\n\u001b[0;32m
884\u001b[0m                                                    \u001b[39mwith\u001b[39;00m
OptionalXlaContext(\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_jit_compile):\n\u001b[1
;32m-->        885\u001b[0m                              result         \u001b[39m=\u001b[39m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_call(\u001b[39m*\u001b[39margs,
\u001b[39m*\u001b[39m\u001b[39m*\u001b[39mkwds)\n\u001b[0;32m               887\u001b[0m
new_tracing_count                                           \u001b[39m=\u001b[39m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39mexperimental_get_tracing_count()\n\u001b[0;
32m             888\u001b[0m   without_tracing   \u001b[39m=\u001b[39m   (tracing_count
\u001b[39m==\u001b[39m new_tracing_count)\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\pytho
n\\eager\\def_function.py:917\u001b[0m, in \u001b[0;36mFunction._call\u001b[1;34m(self,
*args,      **kwds)\u001b[0m\n\u001b[0;32m                        914\u001b[0m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_lock\u001b[39m.\u001b[39mrelease()\n\u001b
[0;32m    915\u001b[0m   \u001b[39m# In this case we have created variables on the first
call, so we run the\u001b[39;00m\n\u001b[0;32m     916\u001b[0m    \u001b[39m# defunned
version which is guaranteed to never create variables.\u001b[39;00m\n\u001b[1;32m-->
917\u001b[0m                                                \u001b[39mreturn\u001b[39;00m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_stateless_fn(\u001b[39m*\u001b[39margs,
\u001b[39m*\u001b[39m\u001b[39m*\u001b[39mkwds)               \u001b[39m#        pylint:
disable=not-callable\u001b[39;00m\n\u001b[0;32m                                 918\u001b[0m
\u001b[39melif\u001b[39;00m     \u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_stateful_fn
\u001b[39mis\u001b[39;00m                                   \u001b[39mnot\u001b[39;00m
\u001b[39mNone\u001b[39;00m:\n\u001b[0;32m      919\u001b[0m     \u001b[39m# Release the
lock early so that multiple threads can perform the call\u001b[39;00m\n\u001b[0;32m
920\u001b[0m    \u001b[39m# in parallel.\u001b[39;00m\n\u001b[0;32m      921\u001b[0m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_lock\u001b[39m.\u001b[39mrelease()\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\pytho
n\\eager\\function.py:3039\u001b[0m, in \u001b[0;36mFunction.__call__\u001b[1;34m(self,
*args,  **kwargs)\u001b[0m\n\u001b[0;32m      3036\u001b[0m  \u001b[39mwith\u001b[39;00m
```

```
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_lock:\n\u001b[0;32m      3037\u001b[0m
(graph_function,\n\u001b[0;32m           3038\u001b[0m               filtered_flat_args)
\u001b[39m=\u001b[39m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_maybe_define_function(args,
kwargs)\n\u001b[1;32m-> 3039\u001b[0m           \u001b[39mreturn\u001b[39;00m
graph_function\u001b[39m.\u001b[39;49m_call_flat(\n\u001b[0;32m     3040\u001b[0m
filtered_flat_args,
captured_inputs\u001b[39m=\u001b[39;49mgraph_function\u001b[39m.\u001b[39;49mcaptured_in
puts)\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\pytho
n\\eager\\function.py:1963\u001b[0m,                                                in
\u001b[0;36mConcreteFunction._call_flat\u001b[1;34m(self,     args,     captured_inputs,
cancellation_manager)\u001b[0m\n\u001b[0;32m       1959\u001b[0m   possible_gradient_type
\u001b[39m=\u001b[39m
gradients_util\u001b[39m.\u001b[39mPossibleTapeGradientTypes(args)\n\u001b[0;32m
1960\u001b[0m  \u001b[39mif\u001b[39;00m  (possible_gradient_type  \u001b[39m==\u001b[39m
gradients_util\u001b[39m.\u001b[39mPOSSIBLE_GRADIENT_TYPES_NONE\n\u001b[0;32m
1961\u001b[0m          \u001b[39mand\u001b[39;00m  executing_eagerly):\n\u001b[0;32m
1962\u001b[0m        \u001b[39m#   No   tape   is   watching;   skip   to   running   the
function.\u001b[39;00m\n\u001b[1;32m-> 1963\u001b[0m       \u001b[39mreturn\u001b[39;00m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_build_call_outputs(\u001b[39mself\u001b[39
;49m\u001b[39m.\u001b[39;49m_inference_function\u001b[39m.\u001b[39;49mcall(\n\u001b[0;3
2m                 1964\u001b[0m                                                ctx,      args,
cancellation_manager\u001b[39m=\u001b[39;49mcancellation_manager))\n\u001b[0;32m
1965\u001b[0m                    forward_backward                    \u001b[39m=\u001b[39m
\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_select_forward_and_backward_functions(\n\u
001b[0;32m       1966\u001b[0m           args,\n\u001b[0;32m       1967\u001b[0m
possible_gradient_type,\n\u001b[0;32m                1968\u001b[0m
executing_eagerly)\n\u001b[0;32m     1969\u001b[0m  forward_function,  args_with_tangents
\u001b[39m=\u001b[39m forward_backward\u001b[39m.\u001b[39mforward()\n",
"File
\u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\pytho
n\\eager\\function.py:591\u001b[0m,                                                 in
\u001b[0;36m_EagerDefinedFunction.call\u001b[1;34m(self,           ctx,           args,
cancellation_manager)\u001b[0m\n\u001b[0;32m     589\u001b[0m \u001b[39mwith\u001b[39;00m
_InterpolateFunctionError(\u001b[39mself\u001b[39m):\n\u001b[0;32m       590\u001b[0m
\u001b[39mif\u001b[39;00m         cancellation_manager         \u001b[39mis\u001b[39;00m
\u001b[39mNone\u001b[39;00m:\n\u001b[1;32m-->    591\u001b[0m                    outputs
\u001b[39m=\u001b[39m    execute\u001b[39m.\u001b[39;49mexecute(\n\u001b[0;32m
592\u001b[0m
\u001b[39mstr\u001b[39;49m(\u001b[39mself\u001b[39;49m\u001b[39m.\u001b[39;49msignature\
u001b[39m.\u001b[39;49mname),\n\u001b[0;32m           593\u001b[0m
num_outputs\u001b[39m=\u001b[39;49m\u001b[39mself\u001b[39;49m\u001b[39m.\u001b[39;49m_n
um_outputs,\n\u001b[0;32m                 594\u001b[0m
inputs\u001b[39m=\u001b[39;49margs,\n\u001b[0;32m         595\u001b[0m
attrs\u001b[39m=\u001b[39;49mattrs,\n\u001b[0;32m         596\u001b[0m
ctx\u001b[39m=\u001b[39;49mctx)\n\u001b[0;32m                    597\u001b[0m
\u001b[39melse\u001b[39;00m:\n\u001b[0;32m            598\u001b[0m                outputs
\u001b[39m=\u001b[39m
execute\u001b[39m.\u001b[39mexecute_with_cancellation(\n\u001b[0;32m    599\u001b[0m

\u001b[39mstr\u001b[39m(\u001b[39mself\u001b[39m\u001b[39m.\u001b[39msignature\u001b[39m
.\u001b[39mname),\n\u001b[0;32m              600\u001b[0m
```

```
num_outputs\u001b[39m=\u001b[39m\u001b[39mself\u001b[39m\u001b[39m.\u001b[39m_num_outputs,\n\u001b[1;32m      (...)\u001b[0m\n\u001b[0;32m        603\u001b[0m ctx\u001b[39m=\u001b[39mctx,\n\u001b[0;32m          604\u001b[0m cancellation_manager\u001b[39m=\u001b[39mcancellation_manager)\n",
"File \u001b[1;32mc:\\Users\\techi\\anaconda3\\envs\\ml\\lib\\site-packages\\tensorflow\\python\\eager\\execute.py:59\u001b[0m,    in    \u001b[0;36mquick_execute\u001b[1;34m(op_name, num_outputs,  inputs,  attrs,  ctx,  name)\u001b[0m\n\u001b[0;32m          57\u001b[0m \u001b[39mtry\u001b[39;00m:\n\u001b[0;32m                              58\u001b[0m ctx\u001b[39m.\u001b[39mensure_initialized()\n\u001b[1;32m--->  59\u001b[0m      tensors \u001b[39m=\u001b[39m pywrap_tfe\u001b[39m.\u001b[39;49mTFE_Py_Execute(ctx\u001b[39m.\u001b[39m_handle, device_name, op_name,\n\u001b[0;32m     60\u001b[0m  inputs, attrs, num_outputs)\n\u001b[0;32m     61\u001b[0m \u001b[39mexcept\u001b[39;00m core\u001b[39m.\u001b[39m_NotOkStatusException                  \u001b[39mas\u001b[39;00m e:\n\u001b[0;32m                  62\u001b[0m           \u001b[39mif\u001b[39;00m   name \u001b[39mis\u001b[39;00m \u001b[39mnot\u001b[39;00m \u001b[39mNone\u001b[39;00m:\n",
"\u001b[1;31mKeyboardInterrupt\u001b[0m: "
]
}
],
"source": [
"history  =  model.fit(train_generator,    epochs=30,  validation_data=valid_generator, verbose=1)\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"import numpy as np\n",
"from scipy import signal\n",
"import matplotlib.pyplot as plt\n",
"\n",
"def plot_power_vs_freq(channel, eeg, fs=200):\n",
"    \"\"\"Extract power (in dB) vs freq graph from a specific channel of EEG data.\n",
"    \n",
"    Args:\n",
"         channel (int): The channel number (0-indexed) for which the power spectrum is to be computed.\n",
"        eeg (ndarray): The EEG data array of shape (num_channels, num_samples).\n",
"        fs (float): The sampling frequency in Hz (default: 200 Hz).\n",
"        \n",
"    Returns:\n",
"        None (displays a plot of power vs frequency).\n",
"    \"\"\"\n",
"    # Select the specific channel of EEG data\n",
"    eeg_channel = eeg[channel, :]\n",
"    \n",
"    # Compute the power spectrum density using Welch's method\n",
"    f, psd = signal.welch(eeg_channel, fs=fs, nperseg=fs*2, nfft=fs*8)\n",
"    \n",
```

```
"    # Convert power to dB scale\n",
"    psd_db = 10 * np.log10(psd)\n",
"    \n",
"    # Plot the power vs frequency graph\n",
"    plt.plot(f, psd_db)\n",
"    plt.title(f\"Power vs Frequency for Channel {channel}\")\n",
"    plt.xlabel(\"Frequency (Hz)\")\n",
"    plt.ylabel(\"Power Spectral Density (dB/Hz)\")\n",
"    plt.show()\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"# Example usage\n",
"\n",
"plot_power_vs_freq(4, EEG)  # plot power vs frequency for channel 4\n"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.16"
},
"orig_nbformat": 4,
"vscode": {
"interpreter": {
"hash": "0bd6827e5b9b024a8afcecbb4b32b3f39bbe94aa5ba060866c09f0f3ec848126"
}
}
},
"nbformat": 4,
"nbformat_minor": 2
}
```

File: epilepsy.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [],
"source": [
"# For example, here's several helpful packages to load\n",
"\n",
"import numpy as np # linear algebra\n",
"import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)\n",
"import matplotlib.pyplot as plt\n",
"import seaborn as sn\n",
"import warnings\n",
"warnings.filterwarnings('ignore')\n",
"\n",
"\n"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>X1</th>\n",
"      <th>X2</th>\n",
"      <th>X3</th>\n",
"      <th>X4</th>\n",
"      <th>X5</th>\n",
"      <th>X6</th>\n",
```

```
"      <th>X7</th>\n",
"      <th>X8</th>\n",
"      <th>X9</th>\n",
"      <th>X10</th>\n",
"      <th>...</th>\n",
"      <th>X170</th>\n",
"      <th>X171</th>\n",
"      <th>X172</th>\n",
"      <th>X173</th>\n",
"      <th>X174</th>\n",
"      <th>X175</th>\n",
"      <th>X176</th>\n",
"      <th>X177</th>\n",
"      <th>X178</th>\n",
"      <th>y</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>135</td>\n",
"      <td>190</td>\n",
"      <td>229</td>\n",
"      <td>223</td>\n",
"      <td>192</td>\n",
"      <td>125</td>\n",
"      <td>55</td>\n",
"      <td>-9</td>\n",
"      <td>-33</td>\n",
"      <td>-38</td>\n",
"      <td>...</td>\n",
"      <td>-17</td>\n",
"      <td>-15</td>\n",
"      <td>-31</td>\n",
"      <td>-77</td>\n",
"      <td>-103</td>\n",
"      <td>-127</td>\n",
"      <td>-116</td>\n",
"      <td>-83</td>\n",
"      <td>-51</td>\n",
"      <td>4</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>386</td>\n",
"      <td>382</td>\n",
"      <td>356</td>\n",
"      <td>331</td>\n",
"      <td>320</td>\n",
"      <td>315</td>\n",
"      <td>307</td>\n",
"      <td>272</td>\n",
"      <td>244</td>\n",
"      <td>232</td>\n",
```

```
"      <td>...</td>\n",
"      <td>164</td>\n",
"      <td>150</td>\n",
"      <td>146</td>\n",
"      <td>152</td>\n",
"      <td>157</td>\n",
"      <td>156</td>\n",
"      <td>154</td>\n",
"      <td>143</td>\n",
"      <td>129</td>\n",
"      <td>1</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>-32</td>\n",
"      <td>-39</td>\n",
"      <td>-47</td>\n",
"      <td>-37</td>\n",
"      <td>-32</td>\n",
"      <td>-36</td>\n",
"      <td>-57</td>\n",
"      <td>-73</td>\n",
"      <td>-85</td>\n",
"      <td>-94</td>\n",
"      <td>...</td>\n",
"      <td>57</td>\n",
"      <td>64</td>\n",
"      <td>48</td>\n",
"      <td>19</td>\n",
"      <td>-12</td>\n",
"      <td>-30</td>\n",
"      <td>-35</td>\n",
"      <td>-35</td>\n",
"      <td>-36</td>\n",
"      <td>5</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>-105</td>\n",
"      <td>-101</td>\n",
"      <td>-96</td>\n",
"      <td>-92</td>\n",
"      <td>-89</td>\n",
"      <td>-95</td>\n",
"      <td>-102</td>\n",
"      <td>-100</td>\n",
"      <td>-87</td>\n",
"      <td>-79</td>\n",
"      <td>...</td>\n",
"      <td>-82</td>\n",
"      <td>-81</td>\n",
"      <td>-80</td>\n",
"      <td>-77</td>\n",
"      <td>-85</td>\n",
```

```
"      <td>-77</td>\n",
"      <td>-72</td>\n",
"      <td>-69</td>\n",
"      <td>-65</td>\n",
"      <td>5</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>-9</td>\n",
"      <td>-65</td>\n",
"      <td>-98</td>\n",
"      <td>-102</td>\n",
"      <td>-78</td>\n",
"      <td>-48</td>\n",
"      <td>-16</td>\n",
"      <td>0</td>\n",
"      <td>-21</td>\n",
"      <td>-59</td>\n",
"      <td>...</td>\n",
"      <td>4</td>\n",
"      <td>2</td>\n",
"      <td>-12</td>\n",
"      <td>-32</td>\n",
"      <td>-41</td>\n",
"      <td>-65</td>\n",
"      <td>-83</td>\n",
"      <td>-89</td>\n",
"      <td>-73</td>\n",
"      <td>5</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>5 rows × 179 columns</p>\n",
"</div>"
],
"text/plain": [
"    X1   X2   X3   X4   X5   X6   X7   X8   X9  X10  ...  X170  X171  X172  \\\n",
"0  135  190  229  223  192  125   55   -9  -33  -38  ...   -17   -15   -31   \n",
"1  386  382  356  331  320  315  307  272  244  232  ...   164   150   146   \n",
"2  -32  -39  -47  -37  -32  -36  -57  -73  -85  -94  ...    57    64    48   \n",
"3 -105 -101  -96  -92  -89  -95 -102 -100  -87  -79  ...   -82   -81   -80   \n",
"4   -9  -65  -98 -102  -78  -48  -16    0  -21  -59  ...     4     2   -12   \n",
"\n",
"   X173  X174  X175  X176  X177  X178  y  \n",
"0   -77  -103  -127  -116   -83   -51  4  \n",
"1   152   157   156   154   143   129  1  \n",
"2    19   -12   -30   -35   -35   -36  5  \n",
"3   -77   -85   -77   -72   -69   -65  5  \n",
"4   -32   -41   -65   -83   -89   -73  5  \n",
"\n",
"[5 rows x 179 columns]"
]
},
"execution_count": 4,
```

```
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"ESR = pd.read_csv('Epileptic Seizure Recognition.csv')\n",
"ESR = ESR.drop(columns = ESR.columns[0]) \n",
"ESR.head()"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0    9200\n",
"1    2300\n",
"Name: y, dtype: int64"
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"cols = ESR.columns\n",
"tgt = ESR.y\n",
"tgt[tgt > 1] = 0\n",
"tgt.value_counts()"
]
},
{
"cell_type": "code",
"execution_count": 12,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"The number of trials for the non-seizure class is: 9200\n",
"The number of trials for the seizure class is: 2300\n"
]
}
],
"source": [
"non_seizure, seizure = tgt.value_counts()\n",
"print('The number of trials for the non-seizure class is:', non_seizure)\n",
"print('The number of trials for the seizure class is:', seizure)"
]
```

```
},
{
"cell_type": "code",
"execution_count": 13,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"0"
]
},
"execution_count": 13,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"ESR.isnull().sum().sum()\n"
]
},
{
"cell_type": "code",
"execution_count": 14,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>X1</th>\n",
"      <th>X2</th>\n",
"      <th>X3</th>\n",
"      <th>X4</th>\n",
"      <th>X5</th>\n",
"      <th>X6</th>\n",
"      <th>X7</th>\n",
```

```
"      <th>X8</th>\n",
"      <th>X9</th>\n",
"      <th>X10</th>\n",
"      <th>...</th>\n",
"      <th>X170</th>\n",
"      <th>X171</th>\n",
"      <th>X172</th>\n",
"      <th>X173</th>\n",
"      <th>X174</th>\n",
"      <th>X175</th>\n",
"      <th>X176</th>\n",
"      <th>X177</th>\n",
"      <th>X178</th>\n",
"      <th>y</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.00000</td>\n",
"      <td>11500.00000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>...</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"      <td>11500.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>mean</th>\n",
"      <td>-11.581391</td>\n",
"      <td>-10.911565</td>\n",
"      <td>-10.187130</td>\n",
"      <td>-9.143043</td>\n",
"      <td>-8.009739</td>\n",
"      <td>-7.003478</td>\n",
"      <td>-6.502087</td>\n",
"      <td>-6.68713</td>\n",
"      <td>-6.55800</td>\n",
"      <td>-6.168435</td>\n",
"      <td>...</td>\n",
```

```
"      <td>-10.145739</td>\n",
"      <td>-11.630348</td>\n",
"      <td>-12.943478</td>\n",
"      <td>-13.668870</td>\n",
"      <td>-13.363304</td>\n",
"      <td>-13.045043</td>\n",
"      <td>-12.705130</td>\n",
"      <td>-12.426000</td>\n",
"      <td>-12.195652</td>\n",
"      <td>0.200000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>std</th>\n",
"      <td>165.626284</td>\n",
"      <td>166.059609</td>\n",
"      <td>163.524317</td>\n",
"      <td>161.269041</td>\n",
"      <td>160.998007</td>\n",
"      <td>161.328725</td>\n",
"      <td>161.467837</td>\n",
"      <td>162.11912</td>\n",
"      <td>162.03336</td>\n",
"      <td>160.436352</td>\n",
"      <td>...</td>\n",
"      <td>164.652883</td>\n",
"      <td>166.149790</td>\n",
"      <td>168.554058</td>\n",
"      <td>168.556486</td>\n",
"      <td>167.257290</td>\n",
"      <td>164.241019</td>\n",
"      <td>162.895832</td>\n",
"      <td>162.886311</td>\n",
"      <td>164.852015</td>\n",
"      <td>0.400017</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>min</th>\n",
"      <td>-1839.000000</td>\n",
"      <td>-1838.000000</td>\n",
"      <td>-1835.000000</td>\n",
"      <td>-1845.000000</td>\n",
"      <td>-1791.000000</td>\n",
"      <td>-1757.000000</td>\n",
"      <td>-1832.000000</td>\n",
"      <td>-1778.00000</td>\n",
"      <td>-1840.00000</td>\n",
"      <td>-1867.000000</td>\n",
"      <td>...</td>\n",
"      <td>-1867.000000</td>\n",
"      <td>-1865.000000</td>\n",
"      <td>-1642.000000</td>\n",
"      <td>-1723.000000</td>\n",
"      <td>-1866.000000</td>\n",
"      <td>-1863.000000</td>\n",
```

```
"      <td>-1781.000000</td>\n",
"      <td>-1727.000000</td>\n",
"      <td>-1829.000000</td>\n",
"      <td>0.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>25%</th>\n",
"      <td>-54.000000</td>\n",
"      <td>-55.000000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>-55.00000</td>\n",
"      <td>-55.00000</td>\n",
"      <td>-54.000000</td>\n",
"      <td>...</td>\n",
"      <td>-55.000000</td>\n",
"      <td>-56.000000</td>\n",
"      <td>-56.000000</td>\n",
"      <td>-56.000000</td>\n",
"      <td>-55.000000</td>\n",
"      <td>-56.000000</td>\n",
"      <td>-55.000000</td>\n",
"      <td>-55.000000</td>\n",
"      <td>-55.000000</td>\n",
"      <td>0.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>50%</th>\n",
"      <td>-8.000000</td>\n",
"      <td>-8.000000</td>\n",
"      <td>-7.000000</td>\n",
"      <td>-8.000000</td>\n",
"      <td>-8.000000</td>\n",
"      <td>-8.000000</td>\n",
"      <td>-8.000000</td>\n",
"      <td>-8.00000</td>\n",
"      <td>-7.00000</td>\n",
"      <td>-7.000000</td>\n",
"      <td>...</td>\n",
"      <td>-9.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>-9.000000</td>\n",
"      <td>-9.000000</td>\n",
"      <td>-9.000000</td>\n",
"      <td>-9.000000</td>\n",
"      <td>0.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
```

```
"      <th>75%</th>\n",
"      <td>34.000000</td>\n",
"      <td>35.000000</td>\n",
"      <td>36.000000</td>\n",
"      <td>36.000000</td>\n",
"      <td>35.000000</td>\n",
"      <td>36.000000</td>\n",
"      <td>35.000000</td>\n",
"      <td>36.00000</td>\n",
"      <td>36.00000</td>\n",
"      <td>35.250000</td>\n",
"      <td>...</td>\n",
"      <td>34.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>33.000000</td>\n",
"      <td>33.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>34.000000</td>\n",
"      <td>0.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>max</th>\n",
"      <td>1726.000000</td>\n",
"      <td>1713.000000</td>\n",
"      <td>1697.000000</td>\n",
"      <td>1612.000000</td>\n",
"      <td>1518.000000</td>\n",
"      <td>1816.000000</td>\n",
"      <td>2047.000000</td>\n",
"      <td>2047.00000</td>\n",
"      <td>2047.00000</td>\n",
"      <td>2047.000000</td>\n",
"      <td>...</td>\n",
"      <td>1777.000000</td>\n",
"      <td>1472.000000</td>\n",
"      <td>1319.000000</td>\n",
"      <td>1436.000000</td>\n",
"      <td>1733.000000</td>\n",
"      <td>1958.000000</td>\n",
"      <td>2047.000000</td>\n",
"      <td>2047.000000</td>\n",
"      <td>1915.000000</td>\n",
"      <td>1.000000</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>8 rows × 179 columns</p>\n",
"</div>"
],
"text/plain": [
"                X1            X2            X3            X4            X5  \\\n",
```

```
"count  11500.000000  11500.000000  11500.000000  11500.000000  11500.000000   \n",
"mean      -11.581391    -10.911565    -10.187130     -9.143043     -8.009739   \n",
"std       165.626284    166.059609    163.524317    161.269041    160.998007   \n",
"min     -1839.000000  -1838.000000  -1835.000000  -1845.000000  -1791.000000   \n",
"25%       -54.000000    -55.000000    -54.000000    -54.000000    -54.000000   \n",
"50%        -8.000000     -8.000000     -7.000000     -8.000000     -8.000000   \n",
"75%        34.000000     35.000000     36.000000     36.000000     35.000000   \n",
"max      1726.000000   1713.000000   1697.000000   1612.000000   1518.000000   \n",
"\n",
"                 X6            X7           X8           X9           X10  \\\n",
"count  11500.000000  11500.000000  11500.00000  11500.00000  11500.000000   \n",
"mean       -7.003478     -6.502087     -6.68713     -6.55800     -6.168435   \n",
"std       161.328725    161.467837    162.11912    162.03336    160.436352   \n",
"min     -1757.000000  -1832.000000  -1778.00000  -1840.00000  -1867.000000   \n",
"25%       -54.000000    -54.000000    -55.00000    -55.00000    -54.000000   \n",
"50%        -8.000000     -8.000000     -8.00000     -7.00000     -7.000000   \n",
"75%        36.000000     35.000000     36.00000     36.00000     35.250000   \n",
"max      1816.000000   2047.000000   2047.00000   2047.00000   2047.000000   \n",
"\n",
"       ...          X170          X171          X172          X173  \\\n",
"count  ...  11500.000000  11500.000000  11500.000000  11500.000000   \n",
"mean   ...    -10.145739    -11.630348    -12.943478    -13.668870   \n",
"std    ...    164.652883    166.149790    168.554058    168.556486   \n",
"min    ...  -1867.000000  -1865.000000  -1642.000000  -1723.000000   \n",
"25%    ...    -55.000000    -56.000000    -56.000000    -56.000000   \n",
"50%    ...     -9.000000    -10.000000    -10.000000    -10.000000   \n",
"75%    ...     34.000000     34.000000     33.000000     33.000000   \n",
"max    ...   1777.000000   1472.000000   1319.000000   1436.000000   \n",
"\n",
"                X174          X175          X176          X177          X178  \\\n",
"count  11500.000000  11500.000000  11500.000000  11500.000000  11500.000000   \n",
"mean     -13.363304    -13.045043    -12.705130    -12.426000    -12.195652   \n",
"std      167.257290    164.241019    162.895832    162.886311    164.852015   \n",
"min    -1866.000000  -1863.000000  -1781.000000  -1727.000000  -1829.000000   \n",
"25%      -55.000000    -56.000000    -55.000000    -55.000000    -55.000000   \n",
"50%      -10.000000     -9.000000     -9.000000     -9.000000     -9.000000   \n",
"75%       34.000000     34.000000     34.000000     34.000000     34.000000   \n",
"max     1733.000000   1958.000000   2047.000000   2047.000000   1915.000000   \n",
"\n",
"                 y  \n",
"count  11500.000000  \n",
"mean       0.200000  \n",
"std        0.400017  \n",
"min        0.000000  \n",
"25%        0.000000  \n",
"50%        0.000000  \n",
"75%        0.000000  \n",
"max        1.000000  \n",
"\n",
"[8 rows x 179 columns]"
]
},
"execution_count": 14,
"metadata": {},
```

```
"output_type": "execute_result"
}
],
"source": [
"ESR.describe()\n"
]
},
{
"cell_type": "code",
"execution_count": 15,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"(11500,)"
]
},
"execution_count": 15,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"Y = ESR.iloc[:,178].values\n",
"Y.shape"
]
},
{
"cell_type": "code",
"execution_count": 16,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"array([0, 1, 0, ..., 0, 0, 0], dtype=int64)"
]
},
"execution_count": 16,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"Y[Y>1]=0\n",
"Y"
]
},
{
"cell_type": "code",
"execution_count": 17,
"metadata": {},
"outputs": [
```

```
{
"data": {
"text/plain": [
"(11500, 177)"
]
},
"execution_count": 17,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"X = ESR.iloc[:,1:178].values\n",
"X.shape"
]
},
{
"cell_type": "code",
"execution_count": 18,
"metadata": {},
"outputs": [],
"source": [
"from sklearn.model_selection import train_test_split, cross_val_score\n",
"X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)"
]
},
{
"cell_type": "code",
"execution_count": 19,
"metadata": {},
"outputs": [],
"source": [
"from sklearn.preprocessing import StandardScaler\n",
"sc = StandardScaler()\n",
"X_train = sc.fit_transform(X_train)\n",
"X_test = sc.transform(X_test)"
]
},
{
"cell_type": "code",
"execution_count": 20,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Accuracy is: 98.16%\n"
]
}
],
"source": [
"from sklearn.svm import SVC\n",
"clf = SVC()\n",
```

```
"clf.fit(X_train, y_train)\n",
"y_pred_svc = clf.predict(X_test)\n",
"acc_svc = round(clf.score(X_train, y_train) * 100, 2)\n",
"print(\"Accuracy is:\",(str(acc_svc)+'%'))"
]
},
{
"cell_type": "code",
"execution_count": 21,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"[X1      -55\n",
" X2       -9\n",
" X3       52\n",
" X4      111\n",
" X5      135\n",
"        ... \n",
" X173    -62\n",
" X174    -41\n",
" X175    -26\n",
" X176     11\n",
" X177     67\n",
" Name: 6, Length: 177, dtype: int64]"
]
},
"execution_count": 21,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"new_input1 = [ESR.iloc[6, :177]]\n",
"new_input1"
]
},
{
"cell_type": "code",
"execution_count": 22,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"array([1], dtype=int64)"
]
},
"execution_count": 22,
"metadata": {},
"output_type": "execute_result"
}
],
```

```
"source": [
"new_output = clf.predict(new_input1)\n",
"new_output"
]
},
{
"cell_type": "code",
"execution_count": 23,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"\"yes\" you might get seizure be conscious about it\n"
]
}
],
"source": [
"new_output\n",
"if new_output==[1]:\n",
"    print('\"yes\" you might get seizure be conscious about it')\n",
"else:\n",
"    print('You are safe no worries :)')"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.16"
},
"orig_nbformat": 4
},
"nbformat": 4,
"nbformat_minor": 2
}
```

Repository: Hyperparameter-Optimization-CNN-Differential-Evolution

File: README.md

```
# Hyperparameter Optimization of CNN Using Differential Evolution Algorithm
This repository contains the code implementation for the Hyperparameter Optimization of Convolutional Neural Networks for Speech Command Recognition using Differential Evolution

## Introduction

### Differential Evolution:
Differential Evolution (DE) is a population-based optimization algorithm and is a powerful as well as versatile evolutionary algorithm commonly used to solve optimization problems, especially in continuous domains.

DE belongs to the class of evolutionary algorithms that mimic the process of natural selection and evolution. Inspired by the principle of survival of the fittest, DE aims to iteratively improve a population of candidate solutions to find the optimal or near-optimal solution to a given problem.

### Dataset and Problem Statement:
This implementation uses Differential Evolution based Hyperparameter Search for Convolutional Neural Networks for the Google Speech Commands dataset to optimize recognition of speech commands for 8 classes: "down", "go", "left", "no", "right", "stop", "up" and "yes".
The approach is then compared to Optimization of CNN using Genetic Algorithm (GA) and performance of Pre-Trained Deep CNN (DCNN) Models on the same dataset.

## Libraries Used:
- Tensorflow (for CNN Model Architecture, Model Training, Pre-Trained models)
- Librosa (Audio Signal Processing)
- Numpy
- Sklearn (for metrics)
```

File: Speech_Command_Classification_DE.ipynb


```
{
"cells": [
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "vzkLHet6JIPX",
"outputId": "17f4607f-6a1a-41dc-fe00-dfb3e9306004"
},
"outputs": [],
```

```
"source": [
"!pip install python_speech_features"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "khcp0u_yJMQL"
},
"outputs": [],
"source": [
"import os\n",
"import pathlib\n",
"from tensorflow.keras.layers.experimental import preprocessing\n",
"from tensorflow.keras import layers\n",
"from tensorflow.keras import models\n",
"from IPython import display\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"import seaborn as sns\n",
"import tensorflow as tf\n",
"import os\n",
"from scipy.io import wavfile\n",
"import pandas as pd\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"from keras.layers import Conv2D,MaxPooling2D,Flatten,LSTM,BatchNormalization,GlobalAveragePooling2D\n",
"from keras.layers import Dropout,Dense,TimeDistributed\n",
"from keras.models import Sequential\n",
"from keras.applications.resnet import ResNet50\n",
"from keras.utils.np_utils import to_categorical\n",
"from sklearn.utils.class_weight import compute_class_weight\n",
"from tqdm import tqdm\n",
"from python_speech_features import mfcc\n",
"import pickle\n",
"from keras.callbacks import ModelCheckpoint\n",
" \n",
"import librosa as lr"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 36
},
"id": "3wH84isvy6rt",
"outputId": "dd2c97a2-3df5-471a-e573-7714e5ca14f2"
},
"outputs": [],
```

```
"source": [
"import tensorflow as tf\n",
"tf.__version__\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "mPKbqPPdJSEW",
"outputId": "10ecc694-864f-4ac2-ae3b-2e53120c5150"
},
"outputs": [],
"source": [
"data_dir = pathlib.Path('data/mini_speech_commands')\n",
"if not data_dir.exists():\n",
"  tf.keras.utils.get_file(\n",
"      'mini_speech_commands.zip',\n",
"
origin=\"http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands
.zip\",\n",
"      extract=True,\n",
"      cache_dir='.', cache_subdir='data')\n",
" \n",
"commands = np.array(tf.io.gfile.listdir(str(data_dir)))\n",
"commands = commands[commands != 'README.md']\n",
"print('Commands:', commands)\n",
" \n",
" \n",
"filenames = tf.io.gfile.glob(str(data_dir) + '/*/*')\n",
"filenames = tf.random.shuffle(filenames)\n",
"num_samples = len(filenames)\n",
"print('Number of total examples:', num_samples)\n",
"print('Number of examples per label:',\n",
"      len(tf.io.gfile.listdir(str(data_dir/commands[0]))))\n",
"print('Example file tensor:', filenames[0])"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "KfIB-0XhJWRn",
"outputId": "55bbf5e4-299f-4e89-bd87-a3ac626be622"
},
"outputs": [],
"source": [
"train_files = filenames[:6400]\n",
```

```
"val_files = filenames[6400: 6400 + 1000]\n",
"test_files = filenames[-600:]\n",
" \n",
"print('Training set size', len(train_files))\n",
"print('Validation set size', len(val_files))\n",
"print('Test set size', len(test_files))\n",
" \n",
" \n",
"def decode_audio(audio_binary):\n",
"  audio, _ = tf.audio.decode_wav(audio_binary)\n",
"  return tf.squeeze(audio, axis=-1)\n",
" \n",
"def get_label(file_path):\n",
"  parts = tf.strings.split(file_path, os.path.sep)\n",
" \n",
"  # Note: You'll use indexing here instead of tuple unpacking to enable this \n",
"  # to work in a TensorFlow graph.\n",
"  return parts[-2] "
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 1000
},
"id": "Kut1NFEwJZk5",
"outputId": "c3b14a16-52a3-4693-b49c-03289232408b"
},
"outputs": [],
"source": [
"def get_waveform_and_label(file_path):\n",
"  label = get_label(file_path)\n",
"  print(\"label\")\n",
"  print(label)\n",
"  audio_binary = tf.io.read_file(file_path)\n",
"  waveform = decode_audio(audio_binary)\n",
"  print(\"waveform\")\n",
"  print(waveform)\n",
"  return waveform, label\n",
" \n",
" \n",
" \n",
"AUTOTUNE = tf.data.AUTOTUNE\n",
"files_ds = tf.data.Dataset.from_tensor_slices(train_files)\n",
"waveform_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
```

```
"fig, axes = plt.subplots(rows, cols, figsize=(10, 12))\n",
"for i, (audio, label) in enumerate(waveform_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  ax.plot(audio.numpy())\n",
"  ax.set_yticks(np.arange(-1.2, 1.2, 0.2))\n",
"  label = label.numpy().decode('utf-8')\n",
"  ax.set_title(label)\n",
" \n",
"plt.show()\n",
" \n",
" \n",
" \n",
"def get_spectrogram(waveform):\n",
"  # Padding for files with less than 16000 samples\n",
"  zero_padding = tf.zeros([16000] - tf.shape(waveform), dtype=tf.float32)\n",
" \n",
"  # Concatenate audio with padding so that all audio clips will be of the \n",
"  # same length\n",
"  waveform = tf.cast(waveform, tf.float32)\n",
"  equal_length = tf.concat([waveform, zero_padding], 0)\n",
"  spectrogram = tf.signal.stft(\n",
"      equal_length, frame_length=255, frame_step=128)\n",
"      \n",
"  spectrogram = tf.abs(spectrogram)\n",
" \n",
"  return spectrogram\n",
" \n",
" \n",
"for waveform, label in waveform_ds.take(1):\n",
"  label = label.numpy().decode('utf-8')\n",
"  spectrogram = get_spectrogram(waveform)\n",
" \n",
"print('Label:', label)\n",
"print('Waveform shape:', waveform.shape)\n",
"print('Spectrogram shape:', spectrogram.shape)\n",
"print('Audio playback')\n",
"display.display(display.Audio(waveform, rate=16000))\n",
" \n",
" \n",
"def plot_spectrogram(spectrogram, ax):\n",
"  # Convert to frequencies to log scale and transpose so that the time is\n",
"  # represented in the x-axis (columns).\n",
"  log_spec = np.log(spectrogram.T)\n",
"  height = log_spec.shape[0]\n",
"  width = log_spec.shape[1]\n",
"  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)\n",
"  Y = range(height)\n",
"  ax.pcolormesh(X, Y, log_spec)\n",
" \n",
" \n",
"fig, axes = plt.subplots(2, figsize=(12, 8))\n",
"timescale = np.arange(waveform.shape[0])\n",
```

```
"axes[0].plot(timescale, waveform.numpy())\n",
"axes[0].set_title('Waveform')\n",
"axes[0].set_xlim([0, 16000])\n",
"plot_spectrogram(spectrogram.numpy(), axes[1])\n",
"axes[1].set_title('Spectrogram')\n",
"plt.show()\n",
" \n",
" \n",
"def get_spectrogram_and_label_id(audio, label):\n",
"  spectrogram = get_spectrogram(audio)\n",
"  spectrogram = tf.expand_dims(spectrogram, -1)\n",
"  label_id = tf.argmax(label == commands)\n",
"  return spectrogram, label_id\n",
" \n",
" \n",
"spectrogram_ds = waveform_ds.map(\n",
"    get_spectrogram_and_label_id, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
"fig, axes = plt.subplots(rows, cols, figsize=(10, 10))\n",
"for i, (spectrogram, label_id) in enumerate(spectrogram_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  plot_spectrogram(np.squeeze(spectrogram.numpy()), ax)\n",
"  ax.set_title(commands[label_id.numpy()])\n",
"  ax.axis('off')\n",
"  \n",
"plt.show()\n",
" \n",
" \n",
"def preprocess_dataset(files):\n",
"  files_ds = tf.data.Dataset.from_tensor_slices(files)\n",
"  output_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
"  output_ds = output_ds.map(\n",
"      get_spectrogram_and_label_id,  num_parallel_calls=AUTOTUNE)\n",
"  return output_ds\n",
" \n",
" \n",
"train_ds = spectrogram_ds\n",
"val_ds = preprocess_dataset(val_files)\n",
"test_ds = preprocess_dataset(test_files)\n",
"print(\"test_ds\")\n",
"print(type(train_ds)) \n",
" \n",
"batch_size = 64\n",
"train_ds = train_ds.batch(batch_size)\n",
"val_ds = val_ds.batch(batch_size)\n",
"test_ds = test_ds.batch(batch_size) \n",
" \n",
"train_ds = train_ds.cache().prefetch(AUTOTUNE)\n",
```

```
"val_ds = val_ds.cache().prefetch(AUTOTUNE)\n",
"test_ds = test_ds.cache().prefetch(AUTOTUNE)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "8G-TeIYUJdJv",
"outputId": "dbb879d3-5d38-43bd-92f8-b54c1c6450c4"
},
"outputs": [],
"source": [
"iterator = train_ds.__iter__()\n",
"next_element = iterator.get_next()\n",
"pt = next_element[0]\n",
"en = next_element[1]\n",
"print(pt.numpy().shape)\n",
"print(en.numpy())"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "1E9Y8ucDJevt",
"outputId": "bc2bca74-a032-44eb-9ccc-c901b650a0b4"
},
"outputs": [],
"source": [
"iterator1 = val_ds.__iter__()\n",
"next_element1 = iterator1.get_next()\n",
"pt1 = next_element1[0]\n",
"en1 = next_element1[1]\n",
"print(pt1.numpy().shape)\n",
"print(en1.numpy().shape)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "3FseP7xZJgdL",
"outputId": "c3c492c5-433c-4505-9f45-b5fa94689dfd"
},
```

```
"outputs": [],
"source": [
"for spectrogram, _ in spectrogram_ds.take(1):\n",
"  input_shape = spectrogram.shape\n",
"print('Input shape:', input_shape)\n",
"num_labels = len(commands)\n",
"\n",
"norm_layer = preprocessing.Normalization()\n",
"norm_layer.adapt(spectrogram_ds.map(lambda x, _: x))"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "u1EViQPkJiCO"
},
"outputs": [],
"source": [
"from keras import layers\n",
"from keras import models\n",
"from keras.callbacks import EarlyStopping"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "1pxkqTjpJj4T"
},
"outputs": [],
"source": [
"def CNN_model(f1, f2, f3, f4, k, a1, a2, d1, d2, op, ep, fitness):\n",
"  model = models.Sequential([\n",
"    layers.Input(shape=input_shape),\n",
"    preprocessing.Resizing(32, 32), \n",
"    norm_layer,\n",
"  ])\n",
"\n",
"                                                    model.add(Conv2D(input_shape=(32,32,
1),filters=f1,kernel_size=(k,k),padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f1,kernel_size=(k,k),padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d2))\n",
"\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
```

```
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d2))\n",
"\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
"\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
"  model.add(Flatten())\n",
"  model.add(BatchNormalization())\n",
"  model.add(Dense(units=f4,activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(Dense(units=f4,activation=a1))\n",
"  model.add(Dense(units=num_labels, activation=\"softmax\"))\n",
"\n",
"  model.compile(\n",
"    optimizer=op,\n",
"    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
"    metrics=['accuracy'],\n",
"  )\n",
"\n",
"  EPOCHS = ep\n",
"  \n",
"  history = model.fit(\n",
"    train_ds, \n",
"    validation_data=val_ds,  \n",
"    epochs=EPOCHS,\n",
"    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=10),\n",
"  )\n",
"                                  fitness.append((history.history[\"val_accuracy\"][-1],
history.history[\"accuracy\"][-1]))\n",
"  return model, history"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "vPtvheG1Jmfz"
},
"outputs": [],
"source": [
```

```
"# Generating the bounds list for every hyperparameter\n",
"bounds = [\n",
"    [16, 32, 64],                                # f1\n",
"    [32, 64, 128],                               # f2\n",
"    [32, 64, 128],                               # f3\n",
"    [128, 256, 512],                             # f4\n",
"    [3, 5],                                       # k\n",
"    [\"relu\", \"selu\", \"elu\"],                        # a1\n",
"    [\"relu\", \"selu\", \"elu\"],                        # a2\n",
"    (0.1, 0.5),                                  # d1\n",
"    (0.1, 0.5),                                  # d2\n",
"    [\"adamax\", \"adadelta\", \"adam\", \"adagrad\"],  # op\n",
"    [50, 60, 70, 80, 90, 100]                    # ep\n",
"]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "-xhY8ADOJqKN",
"outputId": "5d1473ff-ae15-45ad-96c5-204be71485b7"
},
"outputs": [],
"source": [
"import random\n",
"\n",
"pop_size = 15\n",
"# Initializing a population of size 15\n",
"population     =     [[random.choice(item)     if     type(item)     is     list     else
round(random.uniform(item[0], item[1]), 1) if type(item) is tuple else item for item in
bounds] for _ in range(pop_size)]\n",
"print(\"Population:\")\n",
"for i, hyperparameters in enumerate(population):\n",
"    print(\"Hyperparameters set\", i+1, \":\", hyperparameters)"
]
},
{
"cell_type": "markdown",
"metadata": {
"id": "Ig3_3HeDWX9G"
},
"source": []
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
```

```
"id": "zbdQyQ4HhzDn",
"outputId": "6dcd7c06-d860-4ef1-d5ea-ed919d69dd20"
},
"outputs": [],
"source": [
"population"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "pJM7rSl3J_pc"
},
"outputs": [],
"source": [
"def mutation(individual, population, bounds, mutation_factor=0.8):\n",
"\n",
"    population_copy = population.copy()\n",
"    population_copy.remove(individual)\n",
"    a, b, c = random.sample(population_copy, 3)\n",
"\n",
"        \n",
"    # Compute the difference between b and c\n",
"    diff = [round(b_i - c_i, 1) if isinstance(b_i, (int, float)) else b_i for b_i, c_i
in zip(b, c)]\n",
"    \n",
"     # Mutate the individual x by adding the difference multiplied by the mutation
factor\n",
"    mut_individual = [int(a_i + mutation_factor * d) if i in [0,1,2,3,4,10] and \n",
"                          isinstance(a_i, (int, float)) else random.choice(bounds[i])
\n",
"                          if i in [0,1,2,3,4,10] and not isinstance(a_i, (int, float))
else round(a_i + mutation_factor * d, 1)\n",
"                         if isinstance(a_i, (int, float)) else random.choice(bounds[i])
for i,(a_i, d) in enumerate(zip(a, diff))]\n",
"    \n",
"    # make sure that f1, f2, f3, f4 are within (32, 256) bounds\n",
"    for j in range(4):\n",
"        if mut_individual[j] < 32:\n",
"            mut_individual[j] = 32\n",
"        elif mut_individual[j] > 256:\n",
"            mut_individual[j] = 256\n",
"    # Make sure dropout rate stays between (0.1, 0.5)\n",
"    for j in [7,8]:\n",
"        if mut_individual[j] <= 0:\n",
"            mut_individual[j] = 0.1\n",
"        elif mut_individual[j] >= 0.5:\n",
"            mut_individual[j] = 0.5\n",
"            \n",
"    if mut_individual[4] < 3:\n",
"        mut_individual[4] = 3\n",
"            \n",
"            \n",
```

```
"\n",
"    # Min 50 epochs\n",
"    if mut_individual[10] < 50:\n",
"        mut_individual[10] = 50\n",
"    return mut_individual"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "OylLdZatJ5Mn"
},
"outputs": [],
"source": [
"def recombination(individual, population, bounds, CR=0.9):\n",
"    # Mutate the individual first\n",
"    new_individual = mutation(individual, population, bounds)\n",
"    \n",
"     # Pick a random index R in range 1 to n where n is the dimensionality of the problem being optimized.\n",
"    R = random.randint(1, len(bounds))\n",
"    \n",
"    # Compute the agent's potentially new position\n",
"    y = []\n",
"    for i in range(len(bounds)):\n",
"        # Pick a uniformly distributed random number r_i in range(0,1)\n",
"        r = random.uniform(0, 1)\n",
"        if (r < CR) or (i == R):\n",
"            y_i = new_individual[i]\n",
"        else:\n",
"            y_i = individual[i]\n",
"        y.append(y_i)\n",
"    \n",
"     # If f(y)>=f(x) then replace the agent x in the population with the improved or equal candidate solution y\n",
"\n",
"    fitness_y = []\n",
"    fitness_x = []\n",
"    print(y)\n",
"    print(individual)\n",
"    CNN_model(*y, fitness_y)\n",
"    CNN_model(*individual, fitness_x)\n",
"\n",
"    # Comparing based on validation accuracy\n",
"    if fitness_y[0][0] >= fitness_x[0][0]:\n",
"        return y\n",
"    else:\n",
"        return individual"
]
},
{
"cell_type": "code",
"execution_count": null,
```

```
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "s1QMmNFVlUW5",
"outputId": "9bea0bad-3d62-485c-f356-373946af240f"
},
"outputs": [],
"source": [
"from tensorflow.compat.v1 import ConfigProto\n",
"from tensorflow.compat.v1 import InteractiveSession\n",
"\n",
"config = ConfigProto()\n",
"config.gpu_options.allow_growth = True\n",
"session = InteractiveSession(config=config)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "mJfyGsBuyAng",
"outputId": "7985e388-f60c-418a-ee4b-f9e03fe95347"
},
"outputs": [],
"source": [
"iterations = 10\n",
"\n",
"with open(\"generation_info.txt\", \"w\") as f:\n",
"    for iterator in range(iterations):  \n",
"        for i in range(len(population)):\n",
"             # The below call to recombination also has the mutation call within it so
it mutates\n",
"            new_individual = recombination(population[i], population, bounds)\n",
"            population[i] = new_individual\n",
"        print(\"Iteration\", iterator + 1, \"over\")\n",
"        print(\"Current population: \", population)\n",
"\n",
"\n",
"        fitness_gen = []\n",
"             for item in population: # Do this for the current population of a
generation\n",
"            CNN_model(*item, fitness_gen)\n",
"            max_valaccuracy_index = fitness_gen.index(max(fitness_gen))\n",
"        f.write(\"Generation: \" + str(iterator + 1) + \"\\n\")\n",
"        f.write(str(fitness_gen[max_valaccuracy_index]) + \"\\n\")\n",
"        f.write(str(population[max_valaccuracy_index]) + \"\\n\")"
]
},
{
"cell_type": "code",
```

```
"execution_count": null,
"metadata": {
"id": "vgJF8lwFSVrm"
},
"outputs": [],
"source": [
"best = [36, 183, 256, 56, 6, 'elu', 'relu', 0.1, 0.2, 'adagrad', 114]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "-RAwAqLCSW6U",
"outputId": "262ccace-3cb4-4c4a-b223-a2be2fb2e001"
},
"outputs": [],
"source": [
"fitnessbest = []\n",
"best_model, history = CNN_model(*best, fitnessbest)\n",
"best_model.evaluate(test_ds)"
]
}
],
"metadata": {
"accelerator": "GPU",
"colab": {
"machine_shape": "hm",
"provenance": []
},
"gpuClass": "premium",
"kernelspec": {
"display_name": "Python 3",
"name": "python3"
},
"language_info": {
"name": "python"
}
},
"nbformat": 4,
"nbformat_minor": 0
}
```


File: Speech_Command_Classification_GA.ipynb

```
{
"cells": [
{
"cell_type": "code",
```

```
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "LwmjLot9BfYU",
"outputId": "0fdc84c1-e5d7-4ad8-bda3-999c1893df70"
},
"outputs": [],
"source": [
"!pip install python_speech_features"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 1000
},
"id": "PHdAgO5CqSKr",
"outputId": "0b476678-e2ba-48ee-b1ca-6c438cd94199"
},
"outputs": [],
"source": [
"import os\n",
"import pathlib\n",
"from tensorflow.keras.layers.experimental import preprocessing\n",
"from tensorflow.keras import layers\n",
"from tensorflow.keras import models\n",
"from IPython import display\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"import seaborn as sns\n",
"import tensorflow as tf\n",
"import os\n",
"from scipy.io import wavfile\n",
"import pandas as pd\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"from keras.layers import Conv2D,MaxPooling2D,Flatten,LSTM,BatchNormalization,GlobalAveragePooling2D\n",
"from keras.layers import Dropout,Dense,TimeDistributed\n",
"from keras.models import Sequential\n",
"from keras.applications.resnet import ResNet50\n",
"from keras.utils.np_utils import to_categorical\n",
"from sklearn.utils.class_weight import compute_class_weight\n",
"from tqdm import tqdm\n",
"from python_speech_features import mfcc\n",
"import pickle\n",
"from keras.callbacks import ModelCheckpoint\n",
" \n",
"import librosa as lr\n",
```

```
" \n",
"data_dir = pathlib.Path('data/mini_speech_commands')\n",
"if not data_dir.exists():\n",
"  tf.keras.utils.get_file(\n",
"      'mini_speech_commands.zip',\n",
"
origin=\"http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands
.zip\",\n",
"      extract=True,\n",
"      cache_dir='.', cache_subdir='data')\n",
" \n",
"commands = np.array(tf.io.gfile.listdir(str(data_dir)))\n",
"commands = commands[commands != 'README.md']\n",
"print('Commands:', commands)\n",
" \n",
" \n",
"filenames = tf.io.gfile.glob(str(data_dir) + '/*/*')\n",
"filenames = tf.random.shuffle(filenames)\n",
"num_samples = len(filenames)\n",
"print('Number of total examples:', num_samples)\n",
"print('Number of examples per label:',\n",
"      len(tf.io.gfile.listdir(str(data_dir/commands[0]))))\n",
"print('Example file tensor:', filenames[0])\n",
" \n",
" \n",
" \n",
"train_files = filenames[:6400]\n",
"val_files = filenames[6400: 6400 + 1000]\n",
"test_files = filenames[-600:]\n",
" \n",
"print('Training set size', len(train_files))\n",
"print('Validation set size', len(val_files))\n",
"print('Test set size', len(test_files))\n",
" \n",
" \n",
"def decode_audio(audio_binary):\n",
"  audio, _ = tf.audio.decode_wav(audio_binary)\n",
"  return tf.squeeze(audio, axis=-1)\n",
" \n",
"def get_label(file_path):\n",
"  parts = tf.strings.split(file_path, os.path.sep)\n",
" \n",
"  # Note: You'll use indexing here instead of tuple unpacking to enable this \n",
"  # to work in a TensorFlow graph.\n",
"  return parts[-2] \n",
" \n",
"def get_waveform_and_label(file_path):\n",
"  label = get_label(file_path)\n",
"  print(\"label\")\n",
"  print(label)\n",
"  audio_binary = tf.io.read_file(file_path)\n",
"  waveform = decode_audio(audio_binary)\n",
"  print(\"waveform\")\n",
"  print(waveform)\n",
```

```
"  return waveform, label\n",
" \n",
" \n",
" \n",
"AUTOTUNE = tf.data.AUTOTUNE\n",
"files_ds = tf.data.Dataset.from_tensor_slices(train_files)\n",
"waveform_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
"fig, axes = plt.subplots(rows, cols, figsize=(10, 12))\n",
"for i, (audio, label) in enumerate(waveform_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  ax.plot(audio.numpy())\n",
"  ax.set_yticks(np.arange(-1.2, 1.2, 0.2))\n",
"  label = label.numpy().decode('utf-8')\n",
"  ax.set_title(label)\n",
" \n",
"plt.show()\n",
" \n",
" \n",
" \n",
"def get_spectrogram(waveform):\n",
"  # Padding for files with less than 16000 samples\n",
"  zero_padding = tf.zeros([16000] - tf.shape(waveform), dtype=tf.float32)\n",
" \n",
"  # Concatenate audio with padding so that all audio clips will be of the \n",
"  # same length\n",
"  waveform = tf.cast(waveform, tf.float32)\n",
"  equal_length = tf.concat([waveform, zero_padding], 0)\n",
"  spectrogram = tf.signal.stft(\n",
"      equal_length, frame_length=255, frame_step=128)\n",
"      \n",
"  spectrogram = tf.abs(spectrogram)\n",
" \n",
"  return spectrogram\n",
" \n",
" \n",
"for waveform, label in waveform_ds.take(1):\n",
"  label = label.numpy().decode('utf-8')\n",
"  spectrogram = get_spectrogram(waveform)\n",
" \n",
"print('Label:', label)\n",
"print('Waveform shape:', waveform.shape)\n",
"print('Spectrogram shape:', spectrogram.shape)\n",
"print('Audio playback')\n",
"display.display(display.Audio(waveform, rate=16000))\n",
" \n",
" \n",
```

```
"def plot_spectrogram(spectrogram, ax):\n",
"  # Convert to frequencies to log scale and transpose so that the time is\n",
"  # represented in the x-axis (columns).\n",
"  log_spec = np.log(spectrogram.T)\n",
"  height = log_spec.shape[0]\n",
"  width = log_spec.shape[1]\n",
"  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)\n",
"  Y = range(height)\n",
"  ax.pcolormesh(X, Y, log_spec)\n",
" \n",
" \n",
"fig, axes = plt.subplots(2, figsize=(12, 8))\n",
"timescale = np.arange(waveform.shape[0])\n",
"axes[0].plot(timescale, waveform.numpy())\n",
"axes[0].set_title('Waveform')\n",
"axes[0].set_xlim([0, 16000])\n",
"plot_spectrogram(spectrogram.numpy(), axes[1])\n",
"axes[1].set_title('Spectrogram')\n",
"plt.show()\n",
" \n",
" \n",
"def get_spectrogram_and_label_id(audio, label):\n",
"  spectrogram = get_spectrogram(audio)\n",
"  spectrogram = tf.expand_dims(spectrogram, -1)\n",
"  label_id = tf.argmax(label == commands)\n",
"  return spectrogram, label_id\n",
" \n",
" \n",
"spectrogram_ds = waveform_ds.map(\n",
"    get_spectrogram_and_label_id, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
"fig, axes = plt.subplots(rows, cols, figsize=(10, 10))\n",
"for i, (spectrogram, label_id) in enumerate(spectrogram_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  plot_spectrogram(np.squeeze(spectrogram.numpy()), ax)\n",
"  ax.set_title(commands[label_id.numpy()])\n",
"  ax.axis('off')\n",
"  \n",
"plt.show()\n",
" \n",
" \n",
"def preprocess_dataset(files):\n",
"  files_ds = tf.data.Dataset.from_tensor_slices(files)\n",
"  output_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
"  output_ds = output_ds.map(\n",
"      get_spectrogram_and_label_id,  num_parallel_calls=AUTOTUNE)\n",
"  return output_ds\n",
" \n",
```

```
" \n",
"train_ds = spectrogram_ds\n",
"val_ds = preprocess_dataset(val_files)\n",
"test_ds = preprocess_dataset(test_files)\n",
"print(\"test_ds\")\n",
"print(type(train_ds)) \n",
" \n",
"batch_size = 64\n",
"train_ds = train_ds.batch(batch_size)\n",
"val_ds = val_ds.batch(batch_size)\n",
"test_ds = test_ds.batch(batch_size) \n",
" \n",
"train_ds = train_ds.cache().prefetch(AUTOTUNE)\n",
"val_ds = val_ds.cache().prefetch(AUTOTUNE)\n",
"test_ds = test_ds.cache().prefetch(AUTOTUNE)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "xpX6p3OrpzUx",
"outputId": "71bf1315-5da2-4dd0-aee8-868333351b1b"
},
"outputs": [],
"source": [
"iterator = train_ds.__iter__()\n",
"next_element = iterator.get_next()\n",
"pt = next_element[0]\n",
"en = next_element[1]\n",
"print(pt.numpy().shape)\n",
"print(en.numpy())"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "7pWELHqysCe_",
"outputId": "889a61b3-a51a-43c4-a024-3662e1c9752b"
},
"outputs": [],
"source": [
"iterator1 = val_ds.__iter__()\n",
"next_element1 = iterator1.get_next()\n",
"pt1 = next_element1[0]\n",
"en1 = next_element1[1]\n",
"print(pt1.numpy().shape)\n",
```

```
"print(en1.numpy().shape)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "2GlOJ7jRrGFw"
},
"outputs": [],
"source": []
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "c4mFyDRP5dZL"
},
"outputs": [],
"source": [
"import os\n",
"from scipy.io import wavfile\n",
"import pandas as pd\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"from keras.layers import Conv2D,MaxPooling2D,Flatten,LSTM,BatchNormalization,GlobalAveragePooling2D\n",
"from keras.layers import Dropout,Dense,TimeDistributed\n",
"from keras.models import Sequential\n",
"from keras.utils.np_utils import to_categorical\n",
"from sklearn.utils.class_weight import compute_class_weight\n",
"from tqdm import tqdm\n",
"from python_speech_features import mfcc\n",
"import pickle\n",
"from keras.callbacks import ModelCheckpoint\n",
"\n",
"import librosa as lr"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "GeSanShI671x"
},
"outputs": [],
"source": [
"import tensorflow as tf\n",
"from tensorflow import keras\n",
"import numpy as np\n"
]
},
{
"cell_type": "code",
```

```
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "AIqV5naeKbql",
"outputId": "3f96ef45-68f7-4c10-cf11-dc8be512b539"
},
"outputs": [],
"source": [
"for spectrogram, _ in spectrogram_ds.take(1):\n",
"  input_shape = spectrogram.shape\n",
"print('Input shape:', input_shape)\n",
"num_labels = len(commands)\n",
"\n",
"norm_layer = preprocessing.Normalization()\n",
"norm_layer.adapt(spectrogram_ds.map(lambda x, _: x))"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "ueKjpXWOjJ5o"
},
"outputs": [],
"source": [
"from keras import layers\n",
"from keras import models\n",
"from keras.callbacks import EarlyStopping"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "cEk9qXHW4eXx"
},
"outputs": [],
"source": [
"def CNN_model( f1, f2, f3, f4, k, a1, a2, d1, d2, op, ep):\n",
"  model = models.Sequential([\n",
"    layers.Input(shape=input_shape),\n",
"    preprocessing.Resizing(32, 32), \n",
"    norm_layer,\n",
"  ])\n",
"\n",
"                                                    model.add(Conv2D(input_shape=(32,32,
1),filters=f1,kernel_size=(k,k),padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f1,kernel_size=(k,k),padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
```

```
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d2))\n",
"\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a2))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d2))\n",
"\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f2, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
"\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(Conv2D(filters=f3, kernel_size=(k,k), padding=\"same\", activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(MaxPooling2D(pool_size=(2,2),strides=(1,1)))\n",
"  model.add(Dropout(d1))\n",
"\n",
"  model.add(Flatten())\n",
"  model.add(BatchNormalization())\n",
"  model.add(Dense(units=f4,activation=a1))\n",
"  model.add(BatchNormalization())\n",
"  model.add(Dense(units=f4,activation=a1))\n",
"  model.add(Dense(units=num_labels, activation=\"softmax\"))\n",
"\n",
"  model.compile(\n",
"    optimizer=op,\n",
"    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
"    metrics=['accuracy'],\n",
"  )\n",
"  EPOCHS = ep\n",
"  history = model.fit(\n",
"    train_ds, \n",
"    validation_data=val_ds,  \n",
"    epochs=EPOCHS,\n",
"    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=6),\n",
"  )\n",
"  #store history values in global dic.\n",
"  return model, history\n",
"\n",
"\n"
]
},
{
```

```
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "TZRN4Zkh1Yu_"
},
"outputs": [],
"source": [
"from random import choice\n",
"from random import uniform\n",
"from numpy.random import randint"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "i6oy-UhmslXQ"
},
"outputs": [],
"source": [
"def initialization():  \n",
"  parameters = {}\n",
"  f1 = choice([16, 32, 64])\n",
"  parameters[\"f1\"] = f1\n",
"  f2 = choice([32, 64, 128])\n",
"  parameters[\"f2\"] = f2\n",
"  f3 = choice([32, 64, 128])\n",
"  parameters[\"f3\"] = f3\n",
"  f4 = choice([128, 256, 512])\n",
"  parameters[\"f4\"] = f4\n",
"  k = choice([3,5])\n",
"  parameters[\"k\"] = k\n",
"  a1 = choice([\"relu\", \"selu\", \"elu\"])\n",
"  parameters[\"a1\"] = a1\n",
"  a2 = choice([\"relu\", \"selu\", \"elu\"])\n",
"  parameters[\"a2\"] = a2\n",
"  d1 = round(uniform(0.1, 0.5), 1)\n",
"  parameters[\"d1\"] = d1\n",
"  d2 = round(uniform(0.1, 0.5), 1)\n",
"  parameters[\"d2\"] = d2\n",
"  op = choice([\"adamax\", \"adadelta\", \"adam\", \"adagrad\"])\n",
"  parameters[\"op\"] = op\n",
"  ep = randint(50,100)\n",
"  parameters[\"ep\"] = ep\n",
"  return parameters"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "87ThhBE96zCm"
},
"outputs": [],
```

```
"source": [
"def generate_population(n):\n",
"  population = []\n",
"  for i in range(n):\n",
"    chromosome = initialization()\n",
"    population.append(chromosome)\n",
"  return population"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "SomjG3tQ310Z"
},
"outputs": [],
"source": [
"# Fitness evaluation metric: Classification Accuracy \n",
"def fitness_evaluation(model):\n",
"  metrics = model.evaluate(test_ds)\n",
"  print(f\"metrics:{metrics}\")\n",
"  return metrics[1]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "-d8QyqTQ35O1"
},
"outputs": [],
"source": [
"# Roulette wheel selection method\n",
"def selection(population_fitness):\n",
"  total = sum(population_fitness)a\n",
"  percentage = [round((x/total) * 100) for x in population_fitness]\n",
"  selection_wheel = []\n",
"  for pop_index,num in enumerate(percentage):\n",
"    selection_wheel.extend([pop_index]*num)\n",
"  parent1_ind = choice(selection_wheel)\n",
"  parent2_ind = choice(selection_wheel)\n",
"  return [parent1_ind, parent2_ind]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "9tw9NKnz379Q"
},
"outputs": [],
"source": [
"def crossover(parent1, parent2):\n",
"  child1 = {}\n",
```

```
"  child2 = {}\n",
"\n",
"  child1[\"f1\"] = choice([parent1[\"f1\"], parent2[\"f1\"]])\n",
"  child1[\"f2\"] = choice([parent1[\"f2\"], parent2[\"f2\"]])\n",
"  child1[\"f3\"] = choice([parent1[\"f3\"], parent2[\"f3\"]])\n",
"  child1[\"f4\"] = choice([parent1[\"f4\"], parent2[\"f4\"]])\n",
"\n",
"  child2[\"f1\"] = choice([parent1[\"f1\"], parent2[\"f1\"]])\n",
"  child2[\"f2\"] = choice([parent1[\"f2\"], parent2[\"f2\"]])\n",
"  child2[\"f3\"] = choice([parent1[\"f3\"], parent2[\"f3\"]])\n",
"  child2[\"f4\"] = choice([parent1[\"f4\"], parent2[\"f4\"]])\n",
"\n",
"  child1[\"k\"] = choice([parent1[\"k\"], parent2[\"k\"]])\n",
"  child2[\"k\"] = choice([parent1[\"k\"], parent2[\"k\"]])\n",
"\n",
"  child1[\"a1\"] = parent1[\"a2\"]\n",
"  child2[\"a1\"] = parent2[\"a2\"]\n",
"\n",
"  child1[\"a2\"] = parent2[\"a1\"]\n",
"  child2[\"a2\"] = parent1[\"a1\"]\n",
"\n",
"  child1[\"d1\"] = parent1[\"d1\"]\n",
"  child2[\"d1\"] = parent2[\"d1\"]\n",
"\n",
"  child1[\"d2\"] = parent2[\"d2\"]\n",
"  child2[\"d2\"] = parent1[\"d2\"]\n",
"\n",
"  child1[\"op\"] = parent2[\"op\"]\n",
"  child2[\"op\"] = parent1[\"op\"]\n",
"\n",
"  child1[\"ep\"] = parent1[\"ep\"]\n",
"  child2[\"ep\"] = parent2[\"ep\"]\n",
"  return [child1, child2]"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "OvHJT5d23-AN"
},
"outputs": [],
"source": [
"def mutation(chromosome):\n",
"  flag = randint(0,40)\n",
"  if flag <= 20:\n",
"    chromosome[\"ep\"] += randint(0, 10)\n",
"  return chromosome"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
```

```
"colab": {
"background_save": true,
"base_uri": "https://localhost:8080/"
},
"id": "oHz68ueB4Bv6",
"outputId": "b6dfbd9e-081f-4421-c6b1-617411346c41"
},
"outputs": [],
"source": [
"generations = 12\n",
"threshold = 90\n",
"num_pop = 10\n",
"\n",
"population = generate_population(num_pop)\n",
"acc_best = []\n",
"par_total = []\n",
"acc_total = []\n",
"par_best = []\n",
"for generation in range(generations):\n",
"\n",
"  population_fitness = []\n",
"  per1=[]\n",
"  for chromosome in population:\n",
"    f1 = chromosome[\"f1\"]\n",
"    f2 = chromosome[\"f2\"]\n",
"    f3 = chromosome[\"f3\"]\n",
"    f4 = chromosome[\"f4\"]\n",
"    k = chromosome[\"k\"]\n",
"    a1 = chromosome[\"a1\"]\n",
"    a2 = chromosome[\"a2\"]\n",
"    d1 = chromosome[\"d1\"]\n",
"    d2 = chromosome[\"d2\"]\n",
"    op = chromosome[\"op\"]\n",
"    ep = chromosome[\"ep\"]\n",
"\n",
"    try:\n",
"      model = CNN_model(f1, f2, f3, f4, k, a1, a2, d1, d2, op, ep)\n",
"      acc = fitness_evaluation(model)\n",
"      par_total.append(chromosome)\n",
"      acc_total.append(acc)\n",
"      per1.append(chromosome)\n",
"      print(\"Parameters: \", chromosome)\n",
"      print(\"Accuracy: \", round(acc,3))\n",
"    except:\n",
"      acc=0\n",
"      print(\"Parameters: \", chromosome)\n",
"      print(\"Invalid parameters - Build fail\")\n",
"\n",
"    population_fitness.append(acc)\n",
"  print(population_fitness)\n",
"  parents_ind = selection(population_fitness)\n",
"  parent1 = population[parents_ind[0]]\n",
"  parent2 = population[parents_ind[1]]\n",
"\n",
```

```
"  children = crossover(parent1, parent2)\n",
"  child1 = mutation(children[0])\n",
"  child2 = mutation(children[1])\n",
"\n",
"  population.append(child1)\n",
"  population.append(child2)\n",
"\n",
"  print(\"Generation \", generation+1,\" Outcome: \")\n",
"  if max(population_fitness) >= threshold:\n",
"    print(\"Obtained desired accuracy: \", max(population_fitness))\n",
"    break\n",
"  else:\n",
"          print(\"Maximum   accuracy   in   generation   {}   :   {}\".format(generation+1,
max(population_fitness)))\n",
"    max_pop = max(population_fitness)\n",
"    c1=population_fitness.index(max_pop)\n",
"    par_best.append(per1[c1])\n",
"    acc_best.append(max_pop)\n",
"\n",
"  first_min = min(population_fitness)\n",
"  first_min_ind = population_fitness.index(first_min)\n",
"  population.remove(population[first_min_ind])\n",
"  second_min = min(population_fitness)\n",
"  second_min_ind = population_fitness.index(second_min)\n",
"  population.remove(population[second_min_ind])\n",
"\n",
"print(par_total)\n",
"print(acc_total)\n",
"print(par_best)\n",
"print(acc_best)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "9pACtZPGxVhc"
},
"outputs": [],
"source": [
"var1=len(acc_best)\n",
"acc_best.sort()\n",
"print(acc_best[var1-1])"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "12wploAK1JAa",
"outputId": "1e4f4f4b-b9ac-474d-f204-3dc3cd2291d4"
```

```
},
"outputs": [],
"source": [
"best_model, best_history = CNN_model(*[32, 128, 64, 128, 3, 'selu', 'relu', 0.4, 0.2,
'adam', 94])"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "riQe2Ou93fCe",
"outputId": "d0adce7b-fb5e-488d-e390-140648ad3391"
},
"outputs": [],
"source": [
"best_model.evaluate(test_ds)"
]
}
],
"metadata": {
"accelerator": "GPU",
"colab": {
"machine_shape": "hm",
"provenance": []
},
"gpuClass": "premium",
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.15"
},
"vscode": {
"interpreter": {
"hash": "0bd6827e5b9b024a8afcecbb4b32b3f39bbe94aa5ba060866c09f0f3ec848126"
}
}
},
"nbformat": 4,
```

```
"nbformat_minor": 0
}
```


File: Speech_Command_Classification_PRETRAINED.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "u22n2d1MbjW6",
"outputId": "3e87fb31-c394-4871-f77c-a10ffa99d58f"
},
"outputs": [],
"source": [
"!pip install python_speech_features"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "ZLOltoq7opN3",
"outputId": "5ea550ae-8898-4b05-caed-f990fbc829ce"
},
"outputs": [],
"source": [
"!pip install image-classifiers"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "khcp0u_yJMQL"
},
"outputs": [],
"source": [
"import os\n",
"import pathlib\n",
"from tensorflow.keras.layers.experimental import preprocessing\n",
"from IPython import display\n",
"import matplotlib.pyplot as plt\n",
"import numpy as np\n",
"import seaborn as sns\n",
```

```
"import tensorflow as tf\n",
"import os\n",
"from scipy.io import wavfile\n",
"import pandas as pd\n",
"import matplotlib.pyplot as plt\n",
"from keras.layers import Conv2D,MaxPooling2D,Flatten,LSTM,BatchNormalization,GlobalAveragePooling2D\n",
"from keras.layers import Dropout,Dense,TimeDistributed\n",
"from keras.models import Sequential\n",
"from keras.applications.resnet import ResNet50\n",
"from keras.utils.np_utils import to_categorical\n",
"from sklearn.utils.class_weight import compute_class_weight\n",
"from tqdm import tqdm\n",
"from python_speech_features import mfcc\n",
"import pickle\n",
"from keras.callbacks import ModelCheckpoint\n",
" \n",
"import librosa as lr"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 36
},
"id": "3wH84isvy6rt",
"outputId": "d68dabb6-7c82-4616-8dfb-11c7dae7d46e"
},
"outputs": [],
"source": [
"tf.__version__"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "RZI50B2Wbi0z"
},
"outputs": [],
"source": [
"# Set the random seed for TensorFlow and NumPy\n",
"tf.random.set_seed(1)\n",
"np.random.seed(1)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
```

```
"base_uri": "https://localhost:8080/"
},
"id": "mPKbqPPdJSEW",
"outputId": "2594d7b7-f1b5-49a7-8a96-ded2125dadde"
},
"outputs": [],
"source": [
"data_dir = pathlib.Path('data/mini_speech_commands')\n",
"if not data_dir.exists():\n",
"  tf.keras.utils.get_file(\n",
"      'mini_speech_commands.zip',\n",
"
origin=\"http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands
.zip\",\n",
"      extract=True,\n",
"      cache_dir='.', cache_subdir='data')\n",
" \n",
"commands = np.array(tf.io.gfile.listdir(str(data_dir)))\n",
"commands = commands[commands != 'README.md']\n",
"print('Commands:', commands)\n",
" \n",
" \n",
"filenames = tf.io.gfile.glob(str(data_dir) + '/*/*')\n",
"filenames = tf.random.shuffle(filenames)\n",
"num_samples = len(filenames)\n",
"print('Number of total examples:', num_samples)\n",
"print('Number of examples per label:',\n",
"      len(tf.io.gfile.listdir(str(data_dir/commands[0]))))\n",
"print('Example file tensor:', filenames[0])"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "KfIB-0XhJWRn",
"outputId": "9e16e3da-1485-44d7-e64e-73be7dca491f"
},
"outputs": [],
"source": [
"train_files = filenames[:6400]\n",
"val_files = filenames[6400: 6400 + 1000]\n",
"test_files = filenames[-600:]\n",
" \n",
"print('Training set size', len(train_files))\n",
"print('Validation set size', len(val_files))\n",
"print('Test set size', len(test_files))\n",
" \n",
" \n",
"def decode_audio(audio_binary):\n",
"  audio, _ = tf.audio.decode_wav(audio_binary)\n",
```

```
"  return tf.squeeze(audio, axis=-1)\n",
" \n",
"def get_label(file_path):\n",
"  parts = tf.strings.split(file_path, os.path.sep)\n",
" \n",
"  # Note: You'll use indexing here instead of tuple unpacking to enable this \n",
"  # to work in a TensorFlow graph.\n",
"  return parts[-2] "
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 1000
},
"id": "Kut1NFEwJZk5",
"outputId": "47377c9a-cae7-4c7b-c3f0-3f3897766610"
},
"outputs": [],
"source": [
"def get_waveform_and_label(file_path):\n",
"  label = get_label(file_path)\n",
"  print(\"label\")\n",
"  print(label)\n",
"  audio_binary = tf.io.read_file(file_path)\n",
"  waveform = decode_audio(audio_binary)\n",
"  print(\"waveform\")\n",
"  print(waveform)\n",
"  return waveform, label\n",
" \n",
" \n",
" \n",
"AUTOTUNE = tf.data.AUTOTUNE\n",
"files_ds = tf.data.Dataset.from_tensor_slices(train_files)\n",
"waveform_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
"fig, axes = plt.subplots(rows, cols, figsize=(10, 12))\n",
"for i, (audio, label) in enumerate(waveform_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  ax.plot(audio.numpy())\n",
"  ax.set_yticks(np.arange(-1.2, 1.2, 0.2))\n",
"  label = label.numpy().decode('utf-8')\n",
"  ax.set_title(label)\n",
" \n",
```

```
"plt.show()\n",
" \n",
" \n",
" \n",
"def get_spectrogram(waveform):\n",
"  # Padding for files with less than 16000 samples\n",
"  zero_padding = tf.zeros([16000] - tf.shape(waveform), dtype=tf.float32)\n",
" \n",
"  # Concatenate audio with padding so that all audio clips will be of the \n",
"  # same length\n",
"  waveform = tf.cast(waveform, tf.float32)\n",
"  equal_length = tf.concat([waveform, zero_padding], 0)\n",
"  spectrogram = tf.signal.stft(\n",
"      equal_length, frame_length=255, frame_step=128)\n",
"      \n",
"  spectrogram = tf.abs(spectrogram)\n",
" \n",
"  return spectrogram\n",
" \n",
" \n",
"for waveform, label in waveform_ds.take(1):\n",
"  label = label.numpy().decode('utf-8')\n",
"  spectrogram = get_spectrogram(waveform)\n",
" \n",
"print('Label:', label)\n",
"print('Waveform shape:', waveform.shape)\n",
"print('Spectrogram shape:', spectrogram.shape)\n",
"print('Audio playback')\n",
"display.display(display.Audio(waveform, rate=16000))\n",
" \n",
" \n",
"def plot_spectrogram(spectrogram, ax):\n",
"  # Convert to frequencies to log scale and transpose so that the time is\n",
"  # represented in the x-axis (columns).\n",
"  log_spec = np.log(spectrogram.T)\n",
"  height = log_spec.shape[0]\n",
"  width = log_spec.shape[1]\n",
"  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)\n",
"  Y = range(height)\n",
"  ax.pcolormesh(X, Y, log_spec)\n",
" \n",
" \n",
"fig, axes = plt.subplots(2, figsize=(12, 8))\n",
"timescale = np.arange(waveform.shape[0])\n",
"axes[0].plot(timescale, waveform.numpy())\n",
"axes[0].set_title('Waveform')\n",
"axes[0].set_xlim([0, 16000])\n",
"plot_spectrogram(spectrogram.numpy(), axes[1])\n",
"axes[1].set_title('Spectrogram')\n",
"plt.show()\n",
" \n",
" \n",
"def get_spectrogram_and_label_id(audio, label):\n",
"  spectrogram = get_spectrogram(audio)\n",
```

```
"  spectrogram = tf.expand_dims(spectrogram, -1)\n",
"  label_id = tf.argmax(label == commands)\n",
"  return spectrogram, label_id\n",
" \n",
" \n",
"spectrogram_ds = waveform_ds.map(\n",
"    get_spectrogram_and_label_id, num_parallel_calls=AUTOTUNE)\n",
" \n",
" \n",
"rows = 3\n",
"cols = 3\n",
"n = rows*cols\n",
"fig, axes = plt.subplots(rows, cols, figsize=(10, 10))\n",
"for i, (spectrogram, label_id) in enumerate(spectrogram_ds.take(n)):\n",
"  r = i // cols\n",
"  c = i % cols\n",
"  ax = axes[r][c]\n",
"  plot_spectrogram(np.squeeze(spectrogram.numpy()), ax)\n",
"  ax.set_title(commands[label_id.numpy()])\n",
"  ax.axis('off')\n",
"  \n",
"plt.show()\n",
" \n",
" \n",
"def preprocess_dataset(files):\n",
"  files_ds = tf.data.Dataset.from_tensor_slices(files)\n",
"  output_ds = files_ds.map(get_waveform_and_label, num_parallel_calls=AUTOTUNE)\n",
"  output_ds = output_ds.map(\n",
"      get_spectrogram_and_label_id,  num_parallel_calls=AUTOTUNE)\n",
"  return output_ds\n",
" \n",
" \n",
"train_ds = spectrogram_ds\n",
"val_ds = preprocess_dataset(val_files)\n",
"test_ds = preprocess_dataset(test_files)\n",
"print(\"test_ds\")\n",
"print(type(train_ds)) \n",
" \n",
"batch_size = 64\n",
"train_ds = train_ds.batch(batch_size)\n",
"val_ds = val_ds.batch(batch_size)\n",
"test_ds = test_ds.batch(batch_size) \n",
" \n",
"train_ds = train_ds.cache().prefetch(AUTOTUNE)\n",
"val_ds = val_ds.cache().prefetch(AUTOTUNE)\n",
"test_ds = test_ds.cache().prefetch(AUTOTUNE)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
```

```
},
"id": "8G-TeIYUJdJv",
"outputId": "afe782ae-9dde-4760-8352-b761b603d1dd"
},
"outputs": [],
"source": [
"iterator = train_ds.__iter__()\n",
"next_element = iterator.get_next()\n",
"pt = next_element[0]\n",
"en = next_element[1]\n",
"print(pt.numpy().shape)\n",
"print(en.numpy())"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "1E9Y8ucDJevt",
"outputId": "0588aa48-dac6-4379-f9fa-6a73ec5f7ca3"
},
"outputs": [],
"source": [
"iterator1 = val_ds.__iter__()\n",
"next_element1 = iterator1.get_next()\n",
"pt1 = next_element1[0]\n",
"en1 = next_element1[1]\n",
"print(pt1.numpy().shape)\n",
"print(en1.numpy().shape)"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "3FseP7xZJgdL",
"outputId": "1d2d43a5-8e5c-467b-d147-071eb9fcbe8d"
},
"outputs": [],
"source": [
"for spectrogram, _ in spectrogram_ds.take(1):\n",
"  input_shape = spectrogram.shape\n",
"print('Input shape:', input_shape)\n",
"num_labels = len(commands)\n",
"\n",
"norm_layer = preprocessing.Normalization()\n",
"norm_layer.adapt(spectrogram_ds.map(lambda x, _: x))"
]
```

```
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "u1EViQPkJiCO"
},
"outputs": [],
"source": [
"from keras import layers\n",
"from keras import models\n",
"from keras.callbacks import EarlyStopping"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "EQg-8Udtvjxg",
"outputId": "ee4cc1df-0cf7-4414-ff4a-a8d6d204039e"
},
"outputs": [],
"source": [
"input_shape"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "0Y6U03BWbsCk"
},
"outputs": [],
"source": [
"def preprocess(spectrogram, label):\n",
"    spectrogram = tf.repeat(spectrogram, repeats=3, axis=-1)\n",
"    return spectrogram, label\n",
"\n",
"spectrogram_ds = spectrogram_ds.map(preprocess)\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "Mh5GPwy-fZpD",
"outputId": "f36d9f85-507f-4620-8bca-3db044e14f22"
},
```

```
"outputs": [],
"source": [
"spectrogram_ds"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "91l_5g-ElQ78"
},
"outputs": [],
"source": [
"num_val_samples = 1000\n",
"# Split spectrogram_ds into train_ds and val_ds\n",
"train_ds = spectrogram_ds.skip(num_val_samples)\n",
"val_ds = spectrogram_ds.take(num_val_samples)\n",
"test_split = 0.6\n",
"# Further split val_ds into val_ds and test_ds\n",
"num_test_samples = int(num_val_samples * test_split)\n",
"test_ds = val_ds.take(num_test_samples)\n",
"val_ds = val_ds.skip(num_test_samples)\n",
"\n",
"# Set batch size and shuffle the train_ds\n",
"batch_size = 64\n",
"train_ds = train_ds.shuffle(buffer_size=1000).batch(batch_size)\n",
"test_ds = test_ds.batch(batch_size)\n",
"val_ds = val_ds.batch(batch_size)\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "O7A8qp9cbi04"
},
"outputs": [],
"source": [
"# Define a function to extract the labels from the dataset\n",
"def get_label(spectrogram, label):\n",
"    return label\n",
"\n",
"# Map the get_label function to the train_ds to extract the labels\n",
"train_labels_ds = train_ds.map(get_label)\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "BRBo3IUYbi04",
```

```
"outputId": "44c903bd-fe4b-4604-fc3a-23f4789cbd45"
},
"outputs": [],
"source": [
"test_labels_ds = test_ds.map(get_label)\n",
"test_labels_ds\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"id": "tq6wXBGa0qbR"
},
"outputs": [],
"source": [
"from tensorflow.keras import layers\n",
"from tensorflow.keras import models\n",
"from tensorflow.keras import optimizers\n",
"from tensorflow.keras.applications import VGG19\n",
"from tensorflow.keras.applications import VGG16\n",
"from tensorflow.keras.applications import ResNet50\n",
"from tensorflow.keras.applications import InceptionV3\n",
"from tensorflow.keras.applications import Xception\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "Fjc7Chru6dNL",
"outputId": "07b97980-b2fd-4880-bb73-faf747a5de71"
},
"outputs": [],
"source": [
"# VGG19\n",
"# Define the input shape\n",
"input_shape = (124, 129, 3)\n",
"\n",
"# Define the VGG19 model with pre-trained weights\n",
"base_model = VGG19(weights='imagenet', include_top=False, input_shape=input_shape)\n",
"\n",
"# Freeze all layers in the base model\n",
"for layer in base_model.layers:\n",
"    layer.trainable = False\n",
"\n",
"# Add a custom head to the model\n",
"x = layers.Flatten()(base_model.output)\n",
"x = layers.Dense(256, activation='relu')(x)\n",
"x = layers.Dropout(0.5)(x)\n",
"output = layers.Dense(num_labels, activation='softmax')(x)\n",
```

```
"\n",
"# Compile the model\n",
"vgg19_model = models.Model(inputs=base_model.input, outputs=output)\n",
"vgg19_model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizers.Adam(learning_rate=0.001), metrics=['accuracy'])\n",
"\n",
"print(\"VGG19\")\n",
"# Train the model on the train dataset\n",
"vgg19_model.fit(train_ds,               epochs=50,               validation_data=val_ds,
callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=5))\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "J8djwcRv6dV2",
"outputId": "c55fe70b-9adc-477a-c607-4934c7bd17fe"
},
"outputs": [],
"source": [
"# VGG16\n",
"# Define the input shape\n",
"input_shape = (124, 129, 3)\n",
"\n",
"# Define the VGG16 model with pre-trained weights\n",
"base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)\n",
"\n",
"# Freeze all layers in the base model\n",
"for layer in base_model.layers:\n",
"    layer.trainable = False\n",
"\n",
"# Add a custom head to the model\n",
"x = layers.Flatten()(base_model.output)\n",
"x = layers.Dense(256, activation='relu')(x)\n",
"x = layers.Dropout(0.5)(x)\n",
"output = layers.Dense(num_labels, activation='softmax')(x)\n",
"\n",
"# Compile the model\n",
"vgg16_model = models.Model(inputs=base_model.input, outputs=output)\n",
"vgg16_model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizers.Adam(learning_rate=0.001), metrics=['accuracy'])\n",
"\n",
"print(\"VGG16\")\n",
"# Train the model on the train dataset\n",
"vgg16_model.fit(train_ds,               epochs=50,               validation_data=val_ds,
callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=5))\n"
]
},
{
"cell_type": "code",
```

```
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "5PfH58epgcYb",
"outputId": "441353e8-4997-4382-8d4a-ab4cd985f57b"
},
"outputs": [],
"source": [
"# RESNET50\n",
"# Define the input shape\n",
"input_shape = (124, 129, 3)\n",
"\n",
"# Define the ResNet50 model with pre-trained weights\n",
"base_model          =          ResNet50(weights='imagenet',          include_top=False,
input_shape=input_shape)\n",
"\n",
"# Freeze all layers in the base model\n",
"for layer in base_model.layers:\n",
"    layer.trainable = True\n",
"\n",
"# Add a custom head to the model\n",
"x = layers.GlobalAveragePooling2D()(base_model.output)\n",
"x = layers.Dense(256, activation='relu')(x)\n",
"x = layers.Dropout(0.5)(x)\n",
"output = layers.Dense(num_labels, activation='softmax')(x)\n",
"\n",
"# Compile the model\n",
"resnet_model = models.Model(inputs=base_model.input, outputs=output)\n",
"resnet_model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizers.Adam(learning_rate=0.00015), metrics=['accuracy'])\n",
"\n",
"\n",
"print(\"ResNet50\")\n",
"# Train the model on the train dataset\n",
"resnet_model.fit(train_ds,              epochs=100,              validation_data=val_ds,
callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=1))\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "ViDfepakgdiD",
"outputId": "891349c0-8c52-4997-dc6e-9243b29d0d6c"
},
"outputs": [],
"source": [
"# INCEPTIONV3\n",
"\n",
```

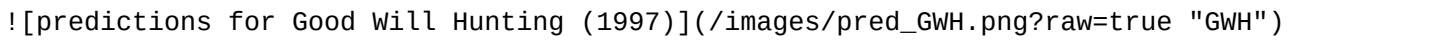
```
"# Define the input shape\n",
"input_shape = (124, 129, 3)\n",
"\n",
"# Define the InceptionV3 model with pre-trained weights\n",
"base_model         =         InceptionV3(weights='imagenet',         include_top=False,
input_shape=input_shape)\n",
"\n",
"# Freeze all layers in the base model\n",
"for layer in base_model.layers:\n",
"    layer.trainable = True\n",
"\n",
"# Add a custom head to the model\n",
"x = layers.GlobalAveragePooling2D()(base_model.output)\n",
"x = layers.Dense(256, activation='relu')(x)\n",
"x = layers.Dropout(0.5)(x)\n",
"output = layers.Dense(num_labels, activation='softmax')(x)\n",
"\n",
"# Compile the model\n",
"inception_model = models.Model(inputs=base_model.input, outputs=output)\n",
"inception_model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizers.Adam(learning_rate=0.00017), metrics=['accuracy'])\n",
"\n",
"print(\"InceptionV3\")\n",
"# Train the model on the train dataset\n",
"inception_model.fit(train_ds,             epochs=100,             validation_data=val_ds,
callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=1))\n"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "zaMxormdgs4J",
"outputId": "b6a5adb9-0e17-46bd-9a4e-bfc69cabfa89"
},
"outputs": [],
"source": [
"# XCEPTION\n",
"\n",
"# Define the input shape\n",
"input_shape = (124, 129, 3)\n",
"\n",
"# Define the Xception model with pre-trained weights\n",
"base_model          =          Xception(weights='imagenet',          include_top=False,
input_shape=input_shape)\n",
"\n",
"# Freeze all layers in the base model\n",
"for layer in base_model.layers:\n",
"    layer.trainable = True\n",
"\n",
"# Add a custom head to the model\n",
```

```
"x = layers.Flatten()(base_model.output)\n",
"x = layers.Dense(256, activation='relu')(x)\n",
"x = layers.Dropout(0.5)(x)\n",
"output = layers.Dense(num_labels, activation='softmax')(x)\n",
"\n",
"# Compile the model\n",
"xception_model = models.Model(inputs=base_model.input, outputs=output)\n",
"xception_model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizers.Adam(learning_rate=0.00001), metrics=['accuracy'])\n",
"\n",
"print(\"Xception\")\n",
"# Train the model on the train dataset\n",
"xception_model.fit(train_ds,              epochs=80,              validation_data=val_ds,
callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=1))\n"
]
}
],
"metadata": {
"accelerator": "GPU",
"colab": {
"machine_shape": "hm",
"provenance": []
},
"gpuClass": "premium",
"kernelspec": {
"display_name": "ml",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.15"
},
"vscode": {
"interpreter": {
"hash": "0bd6827e5b9b024a8afcecbb4b32b3f39bbe94aa5ba060866c09f0f3ec848126"
}
}
},
"nbformat": 4,
"nbformat_minor": 0
}
```

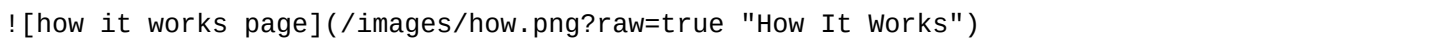
Repository: moviepp

File: README.md

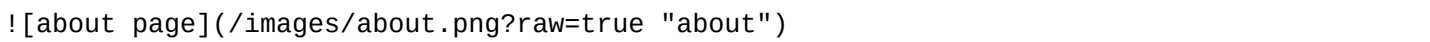
```
MOVIE++
===

Getting recommendations for movies is hard. There's just so much to choose from once you open up IMDb!

So I present my (first!) project, MOVIE++, a movie recommender system using Singular Value Decomposition Matrix Factorization Algorithms and item-based Collaborative Filtering.
It takes a movie name as user input through the form, and spits out 15 recommended movies based on my Machine Learning model.

## Live Demo:
Deployed at: [MOVIE++](https://techie5879.github.io/moviepp/) (Using Github Pages)
The deployed verion uses an SVD model item-based CF recommender system (same as the one detailed in the main code), but it has been trained on a smaller dataset to reduce model file size for deployment. It has been trained using the smaller 100K MovieLens small-latest dataset [MovieLens Small-Latest Permalink](https://grouplens.org/datasets/movielens/latest/). It won't provide the same recommendations as the model trained using the 25M Dataset does, but I've tuned the hyperparameters to give recommendations that seem good enough.

## Technologies Used:

### Frontend:
- ReactJS
- CSS

### Backend:
- Flask API

### Model Training:
- Surprise (Python Library)
- Scikit-learn (Python Library)

## Screenshots:
### Home page:
![Home page](/images/home.png?raw=true "Home Page")
### Autocomplete suggestions for movies:

![suggestions](/images/suggestions.png?raw=true "Home Page_Suggestions")

### Predictions page for The Dark Knight (2008):
---
![predictions for The Dark Knight (2008)](/images/pred_TDK.png?raw=true "TDK")


### Predictions page for Good Will Hunting (1997):
```



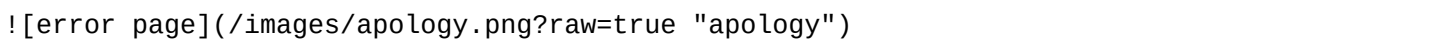
How It Works Page:



About Page:



Error Page (for movie not found):



Usage:

1. User types name of movie, selects a movie from the suggestions list, and clicks the Recommend! button
2. A page with the details of the chosen movie and list of predicted movies with links to their IMDb Pages is displayed.
3. A user can click on the Poster of any movie to be taken to it's respective IMDb Page

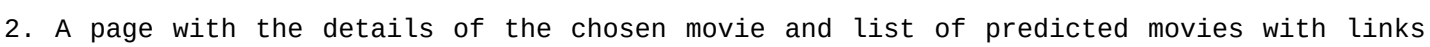
How It Works:

The SVD model was first trained using the Surprise Library in Python. The training was done using the MovieLens 25M dataset provided by GroupLens (permalink: [MovieLens 25M dataset](https://grouplens.org/datasets/movielens/25m/))
It contains approximately 25M ratings across 62423 movies.

The Matrix Factorization method of Singular Value Decomposition is a Dimensionality Reduction method which breaks down the user-product preference matrix into a user-feature and item-feature matrix. This reduces the dimension of the user-product preference space.

Then, SciPy vector cosine distance was used to compute similarity of items (movies) by taking the dot product of the latent feature vectors corresponding to each movie.

The similarity is calculated over all the movies available in the database, and the 15 most similar movies are returned by the Flask API, along with information from the TMDB (The Movie Database) API which has been used to get the poster paths and IMDb IDs of the movies. These are received by the React Frontend and then rendered in the web browser.

Here's a diagram describing the Data Flow:


For running on localhost:
- To get this project up and running on your localhost, you need to have NodeJS installed. Also the requirements from `requirements.txt` must be installed in the environment you're working on. The movielens dataset must be downloaded too and movies.csv, links.csv and ratings.csv are to be placed in the project directory.
- Then `cd` into the repo directory, and use the `svd_train.ipynb` to generate the

model.
- Then use `npm start` to start the frontend, and `flask run` to get the backend running

Challenges Faced and Things Learned:
Since this was my first ML project (and first project in general), getting to this point was quite a challenge. First, I had to learn about how recommender systems worked, types of recommender systems (like Content-based, Collaborative Filtering). It was a continuous process of googling, finding something new, and trying to learn how to implement that! Then I came across the Netflix Prize ([More about Netflix Challenge here](https://en.wikipedia.org/wiki/Netflix_Prize) and [here](https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf)), where a modification of a certain technique called SVD which was quite a bit more efficient than other models grabbed the prize. So naturally I was drawn to learn what it is, and how to implement it. I finally settled on a SVD and item-based Collaborative Filtering model to recommend movies.

Then, the next step was training the model, and there was trouble here too. Although the RMSE error was very respectable, the recommendations didnt look that relevant - there was obscure noise in the data. I used a very primitive method and removed some of the noise by leaving out some obscure movies and their ratings while training. This can probably be improved further to reduce the noise more efficiently. This resulted in very relevant recommendations, so I was satisfied.

Next was the frontend. Even though I had some basic HTML, CSS, Javascript knowledge, I didn't really have any experience. So I decided to learn ReactJS instead of just using HTML Templates with Jinja syntax. I learned how to create multi-page React Websites, conditional routing, use of hooks, states, and requesting and handling data. I also implemented an autocomplete feature in the form.

Deployment of this app to a hosting service is a bit of a difficulty as of now, as the model file generated was of 300+ MB. Hosting such a large model online to get predictions along with a Flask server is quite a problem.
The project has been deployed by making the model smaller by training it on the movielens-small-latest dataset.

File: app.py

```
import os
from dotenv import load_dotenv
from flask import Flask, request
from predict import get_recs
import urllib.request

import json
load_dotenv()

API_KEY = os.getenv("API_KEY")

app = Flask(__name__)

obj = {}
```

```
imdb_result = {}


@app.route('/movies')
def movies():
with open("movies_obj.json") as file:
movies_obj = json.load(file)

movies = movies_obj["title"]
return movies



@app.route('/predictor', methods=["POST"])
def predict():

if request.method == "POST":

movie = request.get_json()
movie_title = movie["title"]
# If movie title isnt empty
if movie_title:
predictions = get_recs(movie_title)
# If movie title in csv list
if predictions:

final = json.loads(predictions)
rec_titles = final["imdbId"]
obj.update(rec_titles)

# If movie title isnt in csv list
for i in range(16):
imdbId = (list(obj.values()))[i]
url                                                                                    =
"https://api.themoviedb.org/3/find/tt{}?api_key={}&language=en-US&external_source=imdb_i
d".format(imdbId, API_KEY)
response = urllib.request.urlopen(url)
data = response.read()
intermediate = json.loads(data)
intermediate["imdbId"] = imdbId
final = json.dumps(intermediate)
imdb_result[i] = json.loads(final)

return imdb_result
else:

return None
```

File: csvtojson.ipynb

```
{
```

```
"cells": [
{
"cell_type": "code",
"execution_count": 1,
"metadata": {},
"outputs": [],
"source": [
"import pandas as pd"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<div>\n",
"<style scoped>\n",
"    .dataframe tbody tr th:only-of-type {\n",
"        vertical-align: middle;\n",
"    }\n",
"\n",
"    .dataframe tbody tr th {\n",
"        vertical-align: top;\n",
"    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>title</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>Toy Story (1995)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>Jumanji (1995)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>Grumpier Old Men (1995)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
```

```
"      <td>Waiting to Exhale (1995)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>Father of the Bride Part II (1995)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>...</th>\n",
"      <td>...</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>62418</th>\n",
"      <td>We (2018)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>62419</th>\n",
"      <td>Window of the Soul (2001)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>62420</th>\n",
"      <td>Bad Poems (2018)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>62421</th>\n",
"      <td>A Girl Thing (2001)</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>62422</th>\n",
"      <td>Women of Devil's Island (1962)</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>62423 rows × 1 columns</p>\n",
"</div>"
],
"text/plain": [
"                                    title\n",
"0                       Toy Story (1995)\n",
"1                         Jumanji (1995)\n",
"2                Grumpier Old Men (1995)\n",
"3               Waiting to Exhale (1995)\n",
"4      Father of the Bride Part II (1995)\n",
"...                                   ...\n",
"62418                          We (2018)\n",
"62419          Window of the Soul (2001)\n",
"62420                   Bad Poems (2018)\n",
"62421                A Girl Thing (2001)\n",
"62422      Women of Devil's Island (1962)\n",
"\n",
"[62423 rows x 1 columns]"
]
},
"execution_count": 11,
"metadata": {},
```

```
"output_type": "execute_result"
}
],
"source": [
"df = pd.read_csv(\"movies.csv\").drop(\"movieId\", axis=1)\n",
"df"
]
},
{
"cell_type": "code",
"execution_count": null,
"metadata": {},
"outputs": [],
"source": [
"df"
]
},
{
"cell_type": "code",
"execution_count": 12,
"metadata": {},
"outputs": [],
"source": [
"df.to_json(r\"D:\\flask_react\\movies_obj.json\")"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "Python 3.9.13 ('tf_gpu')",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.9.13"
},
"orig_nbformat": 4,
"vscode": {
"interpreter": {
"hash": "6727180ec7706371f309a591245f258a273227390eee838732d970e81ce0dc77"
}
}
},
"nbformat": 4,
"nbformat_minor": 2
```

```
}
```

Repository: qiskit-practice

File: 1_Single Qubits.ipynb


```
{
"cells": [
{
"cell_type": "code",
"execution_count": 1,
"metadata": {},
"outputs": [],
"source": [
"from dotenv import load_dotenv\n",
"import os\n",
"from qiskit import *\n",
"from qiskit_ibm_provider import IBMProvider\n",
"from math import *\n",
"import qiskit\n"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Initial Setup for Qiskit, loading accounts and Providers"
]
},
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"{'qiskit-terra':    '0.24.0',    'qiskit-aer':    '0.12.0',    'qiskit-ignis':    None,
'qiskit-ibmq-provider':    '0.20.2',    'qiskit':    '0.43.0',    'qiskit-nature':    None,
'qiskit-finance': None, 'qiskit-optimization': None, 'qiskit-machine-learning': None}"
]
},
"execution_count": 2,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"qiskit.__qiskit_version__"
]
},
{
"cell_type": "code",
```

```
"execution_count": 3,
"metadata": {},
"outputs": [],
"source": [
"load_dotenv()\n",
"IBM_KEY = os.getenv(\"API_KEY\")"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [],
"source": [
"provider = IBMProvider()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Code for initializing a qubit"
]
},
{
"cell_type": "code",
"execution_count": 5,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"QuantumRegister(1, 'q')"
]
},
"execution_count": 5,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"# Initializing a single qubit to |0> and naming the qubit as 'q'\n",
"qr = QuantumRegister(1, \"q\")\n",
"qr"
]
},
{
"cell_type": "code",
"execution_count": 6,
"metadata": {},
"outputs": [
{
"data": {
```

```
"text/plain": [
"Qubit(QuantumRegister(1, 'q'), 0)"
]
},
"execution_count": 6,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"qr[0]"
]
},
{
"cell_type": "code",
"execution_count": 7,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre   style=\"word-wrap:   normal;white-space:   pre;background:   #fff0;line-height:
1.1;font-family:       &amp;quot;Courier       New&amp;quot;,Courier,monospace\">
????????????????????????????????\n",
"q: ? Initialize(0.70711,0.70711) ?\n",
"   ????????????????????????????????</pre>"
],
"text/plain": [
"   ????????????????????????????????\n",
"q: ? Initialize(0.70711,0.70711) ?\n",
"   ????????????????????????????????"
]
},
"execution_count": 7,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"# Initializing a qubit to a desired state\n",
"desired_state = [1/sqrt(2), 1/sqrt(2)]\n",
"qr = QuantumRegister(1, \"q\")\n",
"qc = QuantumCircuit(qr)\n",
"qc.initialize(desired_state, qr[0])\n",
"qc.draw()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"#### Some basic gates are as follows\n",
"\n",
```

```
"$$X = \\begin{bmatrix} 0 & 1 \\\\ 1 & 0 \\end{bmatrix}$$\n",
"$$Y = \\begin{bmatrix} 0 & -i \\\\ i & 0 \\end{bmatrix}$$\n",
"$$Z = \\begin{bmatrix} 1 & 0 \\\\ 0 & -1 \\end{bmatrix}$$\n",
"$$H = \\frac{1}{\\sqrt2}\\begin{bmatrix} 1 & 1 \\\\ 1 & -1 \\end{bmatrix}$$\n",
"$$S = \\begin{bmatrix} 1 & 0 \\\\ 0 & i \\end{bmatrix}$$\n",
"$$T = \\begin{bmatrix} 1 & 0 \\\\ 0 & e^{i\\pi /4} \\end{bmatrix}$$\n",
"\n",
"##### Some useful states that can also be used as basis since they are orthogonal are:\n",
"\n",
"$$| + \\rangle = \\frac{1}{\\sqrt2}(|0\\rangle + |1\\rangle)$$\n",
"$$| - \\rangle = \\frac{1}{\\sqrt2}(|0\\rangle - |1\\rangle)$$"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### A quantum circuit implementing gates H, Z, X\n",
"\n",
"This circuit uses H, Z, and X gates in that order to obtain the state $\\frac{1}{\\sqrt2}(|1\\rangle - |0\\rangle)$ from $|0\\rangle$\n",
"\n",
"First, by default, qr is initialized to $|0\\rangle$, and on application of H, we obtained $\\frac{1}{\\sqrt2}(|0\\rangle + |1\\rangle)$  \n",
"Then application of Z gate makes qr evolve to $\\frac{1}{\\sqrt2}(|0\\rangle - |1\\rangle)$  \n",
"Finally, X acts similar to the classical NOT gate and qr finally evolves to $\\frac{1}{\\sqrt2}(|1\\rangle - |0\\rangle)$"
]
},
{
"cell_type": "code",
"execution_count": 8,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre    style=\"word-wrap:    normal;white-space:    pre;background:    #fff0;line-height: 1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">    ??????????????\n",
"q0: ? H ?? Z ?? X ?\n",
"    ??????????????</pre>"
],
"text/plain": [
"    ??????????????\n",
"q0: ? H ?? Z ?? X ?\n",
"    ??????????????"
]
},
"execution_count": 8,
"metadata": {},
"output_type": "execute_result"
```

```
}
],
"source": [
"qr = QuantumRegister(1)\n",
"qc = QuantumCircuit(qr)\n",
"qc.h(qr)\n",
"qc.z(qr)\n",
"qc.x(qr)\n",
"qc.draw()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Generalization of single qubit gates\n",
"\n",
"$$U_3(\\theta, \\phi, \\lambda) = \\begin{bmatrix} \\cos \\frac{\\theta}{2} & -e^{i\\lambda}\\sin \\frac{\\theta}{2} \\\\[6pt] e^{i\\phi}\\sin \\frac{\\theta}{2} & e^{i(\\phi + \\lambda)\\cos \\frac{\\theta}{2}} \\end{bmatrix}$$ \n",
"We can express X gate as $U_3(\\pi, 0, \\pi)$, H gate as $U_3(\\frac{\\pi}{2}, 0, \\pi)$\n",
"\n",
"#### Other common parametric gates $U_1$ and $U_2$ are as follows:\n",
"$$U_1(\\lambda) = \\begin{bmatrix} 1 & 0 \\\\[6pt] 0 & e^{i\\lambda} \\end{bmatrix} = U_3(0, 0, \\lambda)$$\n",
"\n",
"$$U_2(\\phi, \\lambda) = \\frac{1}{\\sqrt2}\\begin{bmatrix} 1 & -e^{i\\lambda} \\\\[6pt] e^{i\\phi} & e^{i(\\phi + \\lambda)} \\end{bmatrix} = U_3(\\frac{\\pi}{2}, \\phi, \\lambda)$$\n"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Evolving to the same state as above using U1, U2, U3 gates"
]
},
{
"cell_type": "code",
"execution_count": 9,
"metadata": {},
"outputs": [
{
"data": {
"text/plain": [
"<qiskit.circuit.instructionset.InstructionSet at 0x1e847e0d480>"
]
},
"execution_count": 9,
"metadata": {},
```

```
"output_type": "execute_result"
}
],
"source": [
"qr = QuantumRegister(1)\n",
"qc = QuantumCircuit(qr)\n",
"qc.u(pi/2, 0, pi, qr) #u2 gate\n",
"qc.u(0, 0, pi, qr) #u1 gate\n",
"qc.u(pi, 0, pi, qr) #u3 gate"
]
},
{
"cell_type": "code",
"execution_count": 10,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"Statevector([-0.70710678+8.65956056e-17j,  0.70710678+0.00000000e+00j],\n",
"            dims=(2,))\n"
]
}
],
"source": [
"backend = Aer.get_backend('statevector_simulator')\n",
"qjob = execute(qc, backend)\n",
"out_vector = qjob.result().get_statevector()\n",
"print(out_vector)"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Single Qubit Measurement\n",
"\n",
"We flip the qubit qr and measure it 100 times. Since initial state of qr was $|0\\rangle$, flipping it makes its state $|1\\rangle$, so measuring it 100 times should give us the state $|1\\rangle$ 100 times.  \n",
"This measurement is performed in computational basis, and qubit will collapse to one of $|0\\rangle$ or $|1\\rangle$ on measurement"
]
},
{
"cell_type": "code",
"execution_count": 11,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
```

```
"text": [
"{'1': 100}\n"
]
}
],
"source": [
"# Creating Quantum Register, Classical Register, and Quantum Circuit\n",
"qr = QuantumRegister(1)\n",
"cr = ClassicalRegister(1)  # Helps in storing the output of the qubit measurement\n",
"qc = QuantumCircuit(qr, cr)\n",
"\n",
"# Flipping qubit qr and measuring it\n",
"qc.x(qr)\n",
"qc.measure(qr, cr) # Measure the qubit qr and store the ouput in cr\n",
"\n",
"# Executing on backend\n",
"backend = Aer.get_backend('qasm_simulator')\n",
"qjob = execute(qc, backend, shots=100) # shots specifies the number of times the circuit (including measurement) will be repeated\n",
"# Resulting measurement statistic will be stored in result object\n",
"# This mimics measuring 100 qubits in computational basis\n",
"measurement = qjob.result().get_counts()\n",
"print(measurement)"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"#### Measuring in Hadamard Basis\n",
"\n",
"We can measure the state of the qubit in any orthogonal basis. If we want to know whether the output is $|+\\rangle$ or $|-\\rangle$, then we have to measure in the Hadamard Basis: $\\{ |+\\rangle, |-\\rangle \\}$  \n",
"Any state $|\\phi \\rangle = \\alpha |0\\rangle + \\beta |1\\rangle$ can be represented in Hadamard basis using the following:\n",
"$$|\\phi \\rangle = \\frac{(\\alpha + \\beta)}{\\sqrt2} |+\\rangle + \\frac{(\\alpha - \\beta)}{\\sqrt2} |-\\rangle$$\n",
"\n",
"Now, if we apply $H^{\\dagger}$ on $|\\phi\\rangle$, we have:\n",
"$$H^{\\dagger} |\\phi \\rangle = \\frac{(\\alpha + \\beta)}{\\sqrt2} |0\\rangle + \\frac{(\\alpha - \\beta)}{\\sqrt2} |1\\rangle$$\n",
"i.e., $H^{\\dagger}$ maps $|+\\rangle$ to $|0\\rangle$ and $|-\\rangle$ to $|1\\rangle$\n",
"\n",
"So, measuring $H^\\dagger |\\phi \\rangle$ in computational basis is the same as measuring $|\\phi \\rangle$ in Hadamard Basis  \n",
"This is called Transformation of Basis\n",
"\n",
"Note: $H = H^\\dagger$"
]
},
{
```

```
"cell_type": "code",
"execution_count": 17,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"{'1': 488, '0': 536}\n"
]
}
],
"source": [
"qr = QuantumRegister(1)\n",
"cr = ClassicalRegister(1)\n",
"qc = QuantumCircuit(qr, cr)\n",
"\n",
"qc.x(qr)\n",
"# Now we need to measure whether output is one of the Hadamard Basis\n",
"# So first apply H^dagger = H on qr to transform to computational basis\n",
"# In output, 0 corresponds to + and 1 corresponds to -\n",
"\n",
"qc.h(qr)\n",
"qc.measure(qr, cr)\n",
"\n",
"backend = Aer.get_backend('qasm_simulator')\n",
"qjob = execute(qc, backend) # Default shots = 1024\n",
"counts = qjob.result().get_counts()\n",
"print(counts)\n"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"Slight deviation due to randomness, otherwise should have been prob = $\\frac{1}{2}$ for each exactly  \n",
"Note: In output, $|0\\rangle$ corresponds to $|+\\rangle$ and $|1\\rangle$ corresponds to $|-\\rangle$"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "quantum",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
```

```
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.11.3"
},
"orig_nbformat": 4
},
"nbformat": 4,
"nbformat_minor": 2
}
```


File: 2_QTRNG.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 1,
"metadata": {},
"outputs": [],
"source": [
"from dotenv import load_dotenv\n",
"import os\n",
"from qiskit import *\n",
"from qiskit_ibm_provider import IBMProvider\n",
"from math import *\n",
"import qiskit\n",
"\n",
"load_dotenv()\n",
"IBM_KEY = os.getenv(\"API_KEY\")\n",
"provider = IBMProvider()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Quantum True Random Number Generator (QTRNG)\n",
"\n",
"Hadamard gate and measurement in computational basis will give rise to true random sequences (due to inherent property of randomness of quantum states)  \n",
"This isn't possible with a classical scenario with PRNG (Pseudo Random Number Generators)"
]
},
{
"cell_type": "code",
"execution_count": 17,
"metadata": {},
```

```
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"0\n",
"1\n",
"0\n",
"0\n",
"0\n",
"0\n",
"1\n",
"0\n",
"1\n",
"0\n"
]
}
],
"source": [
"for i in range(10):\n",
"    q = QuantumRegister(1)\n",
"    c = ClassicalRegister(1)\n",
"    qc = QuantumCircuit(q, c)\n",
"\n",
"    qc.h(q)\n",
"    qc.measure(q, c)\n",
"\n",
"    backend = Aer.get_backend('qasm_simulator')\n",
"    qjob = execute(qc, backend, shots=1)\n",
"    counts = qjob.result().get_counts()\n",
"    for key in counts.keys():\n",
"        print(key)\n"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "quantum",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.11.3"
},
"orig_nbformat": 4
```

```
},
"nbformat": 4,
"nbformat_minor": 2
}
```

File: 3_Multiple Qubits.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [],
"source": [
"from dotenv import load_dotenv\n",
"import os\n",
"from qiskit import *\n",
"from qiskit_ibm_provider import IBMProvider\n",
"from math import *\n",
"import qiskit\n",
"\n",
"load_dotenv()\n",
"IBM_KEY = os.getenv(\"API_KEY\")\n",
"provider = IBMProvider()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Multiple Qubits\n",
"\n",
"For two qubits, we can have a superposition of the 4 states:\n",
"$$|\\psi \\rangle = \\alpha_0 |00\\rangle + \\alpha_1 |01\\rangle + \\alpha_2 |10\\rangle + \\alpha_3 |11\\rangle $$\n",
"Amplitude of $|\\psi\\rangle$ is denoted as a column vector\n",
"\n",
"$$|\\psi \\rangle = \\begin{bmatrix}\\alpha_0 \\\\ \\alpha_1 \\\\ \\alpha_2 \\\\ \\alpha_3\\end{bmatrix}$$\n",
"\n",
"$|ab\\rangle$ denotes the tensor product $|a\\rangle \\otimes |b\\rangle$ \n",
"For two vectors $|a\\rangle, |b\\rangle$\n",
"\n",
"$$|a\\rangle = \\begin{bmatrix}a_1 \\\\ a_2 \\\\ \\vdots \\\\ a_n\\end{bmatrix}, |b\\rangle = \\begin{bmatrix}b_1 \\\\ b_2 \\\\ \\vdots \\\\ b_m\\end{bmatrix}$$\n",
"$$|a\\rangle \\otimes |b\\rangle = \\begin{bmatrix}a_1 \\cdot b_1 \\\\ \\vdots \\\\ a_1\\cdot b_m \\\\ \\vdots \\\\ a_n\\cdot b_1 \\\\ \\vdots \\\\ a_n\\cdot b_m\\end{bmatrix}$$\n",
"\n",
"Dimension of tensor product is $(n\\cdot m) \\times 1$\n",
```

```
"\n",
"Tensor Product of two matrices $A = (a_{ij})_{m\\times n}, B = (b_{ij})_{p\\times q}$ is given by:\n",
"$$A\\otimes B = \\begin{bmatrix}a_{11}B & a_{12}B & \\dots & a_{1n}B \\\\a_{12}B & a_{22}B & \\dots & a_{2n}B \\\\ \\vdots & & \\ddots & \\vdots \\\\ a_{m1}B & a_{m2}B & \\dots & a_{mn}B\\end{bmatrix}_{mp\\times nq}$$"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"System composed of two qubits of state $|\\phi_1\\rangle, |\\phi_2\\rangle$ can be written as $|\\psi\\rangle = |\\phi_1\\rangle \\otimes |\\phi_2\\rangle$  \n",
"\n",
"Similarly, state of n-qubit system which is composed of n single qubits with states $|\\phi_1 \\rangle, |\\phi_2\\rangle, \\dots |\\phi_n \\rangle$ can be denoted as:\n",
"$$|\\psi \\rangle = |\\phi_1\\rangle \\otimes |\\phi_2\\rangle \\otimes \\dots \\otimes |\\phi_n\\rangle$$\n",
"\n",
"But we cannot always express an arbitrary n-qubit quantum system as a tensor product of n single qubit states, due to Quantum Entanglement"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Quantum Entanglement\n",
"\n",
"A set of qubits whose combined state is $|\\psi \\rangle$ is said to be in a state of entanglement if $|\\psi \\rangle$ cannot be decomposed as $|\\psi \\rangle = |\\phi \\rangle \\otimes |\\xi \\rangle$ where $|\\phi \\rangle$ and $|\\xi \\rangle$ are two independent quantum states.\n",
"\n",
"In $|\\psi\\rangle = \\frac{1}{\\sqrt2}(|00\\rangle + |11\\rangle)$, $|\\psi\\rangle$ cannot be written as a tensor product of two single qubit states - they are in a state of entanglement.  \n",
"However, in the above, if we conduct a measurement of the first qubit, we obtain $|0\\rangle$ with a probability of $\\frac{1}{2}$ and $|1\\rangle$ with a probability of $\\frac{1}{2}$ (same situation if we measure the second qubit first).  \n",
"Suppose we have obtained $|0\\rangle$ on first qubit measurement $\\rightarrow$ we will obtain a measurement of $|0\\rangle$ with a probability of $1$ for the second qubit (as the state of the system has been collapsed to $|00\\rangle$). So from just measuring any one of the qubits, we can learn about the measurement outcome for the remaining qubit.\n",
"\n",
"Measurement of first qubit $\\rightarrow$ Probabilistic  \n",
"Measurement of subsequent second qubit $\\rightarrow$ Determined exactly\n"
]
},
{
```

```
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Bell States or EPR Pairs (Important Two-Qubit States):\n",
"\n",
"$$|\\Phi^+\\rangle = \\frac{1}{\\sqrt2}(|00\\rangle + |11\\rangle)$$\n",
"$$|\\Phi^-\\rangle = \\frac{1}{\\sqrt2}(|00\\rangle - |11\\rangle)$$\n",
"$$|\\Psi^+\\rangle = \\frac{1}{\\sqrt2}(|01\\rangle + |10\\rangle)$$\n",
"$$|\\Psi^-\\rangle = \\frac{1}{\\sqrt2}(|01\\rangle - |10\\rangle)$$\n",
"\n",
"### GHZ States (used in Quantum Error corrections):\n",
"$$|GHZ_\\pm \\rangle = \\frac{1}{\\sqrt2}(|000\\rangle \\pm |111\\rangle)$$"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Multi-Qubit Gates\n",
"\n",
"### Controlled NOT Gate:  \n",
"Analogue to Classical XOR gate $\\rightarrow$ CNOT Gate (Controlled NOT Gate)  \n",
"CNOT Gate has two inputs: \"control\" qubit and \"target\" qubit  \n",
"If control qubit is set to $|1\\rangle$, then flip target qubit, else do nothing (CNOT basically implements X gate on target qubit if controlled qubit is in state $|1\\rangle$)\n",
"$$\\text{CNOT} := \\begin{bmatrix}1 & 0 & 0 & 0 \\\\ 0 & 1 & 0 & 0 \\\\ 0 & 0 & 0 & 1 \\\\ 0 & 0 & 1 & 0\\end{bmatrix}$$\n",
"\n",
"For $|\\psi \\rangle = \\alpha_0 |00\\rangle + \\alpha_1 |01\\rangle + \\alpha_2 |10\\rangle + \\alpha_3 |11\\rangle $, action of CNOT gate can be described as:\n",
"$$\\text{CNOT}|\\psi \\rangle = \\alpha_0 |00\\rangle + \\alpha_1 |01\\rangle + \\alpha_2 |11\\rangle + \\alpha_3 |10\\rangle $$  \n",
"We can state the action of the CNOT Gate in computational basis state of the individual qubits:\n",
"$$\\text{CNOT}|a\\rangle |b\\rangle \\rightarrow |a\\rangle |a \\oplus b\\rangle$$ \n",
"or, we can think of CNOT as a function that acts on two qubits:\n",
"$$\\text{CNOT}(|a\\rangle, |b\\rangle) \\rightarrow (|a\\rangle, |a \\oplus b\\rangle)$$\n",
"\n",
"$\\text{CNOT} + \\text{all single qubit quantum gates} \\rightarrow \\text{can be used to construct any quantum gate}$  \n",
"So, CNOT and all single qubit quantum gates are called universal quantum gates\n",
"\n",
"#### Controlled Hadamard Gate:\n",
"$$C_H = \\begin{bmatrix}1 & 0 & 0 & 0 \\\\ 0 & 1 & 0 & 0 \\\\[6pt] 0 & 0 & \\frac{1}{\\sqrt2} & \\frac{1}{\\sqrt2} \\\\[6pt] 0 & 0 & \\frac{1}{\\sqrt2} & -\\frac{1}{\\sqrt2}\\end{bmatrix}$$\n",
"\n",
"#### Toffoli Gate (CCNOT):\n",
"\n",
```

```
"Extension of CNOT to three qubits, first two are control qubits and the 3rd is target qubit. Target qubit is flipped iff both control qubits are set to 1\n",
"$$CCNOT |a \\rangle | b \\rangle |c\\rangle \\rightarrow |a\\rangle|b\\rangle|c \\oplus (a\\cdot b)\\rangle$$  \n",
"Corresponding unitary matrix for CCNOT gate (operating on 3 qubits, so $8 \\times 8$ matrix)\n",
"$$CCNOT := \\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\\\\n",
"                          0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\\\\n",
"                          0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\\\\n",
"                          0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\\\\n",
"                          0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\\\\n",
"                          0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\\\\n",
"                          0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\\\\n",
"                          0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\\\\n",
"                            \\end{bmatrix}$$"
]
},
{
"cell_type": "code",
"execution_count": 27,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre    style=\"word-wrap:    normal;white-space:    pre;background:    #fff0;line-height: 1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">            ???   \n",
"q89_0: ??????M????\n",
"       ???????????\n",
"q89_1: ? X ?????M?\n",
"       ????? ? ???\n",
"c22: 2/???????????\n",
"             0  1 </pre>"
],
"text/plain": [
"            ???   \n",
"q89_0: ??????M????\n",
"       ???????????\n",
"q89_1: ? X ?????M?\n",
"       ????? ? ???\n",
"c22: 2/???????????\n",
"             0  1 "
]
},
"execution_count": 27,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"q = QuantumRegister(2)\n",
"c = ClassicalRegister(2)\n",
"qc = QuantumCircuit(q, c)\n",
"\n",
```

```
"qc.cx(q[0], q[1])\n",
"qc.measure(q, c)\n",
"\n",
"qc.draw()"
]
},
{
"cell_type": "code",
"execution_count": 29,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre   style=\"word-wrap:   normal;white-space:   pre;background:   #fff0;line-height:
1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">           ???      \n",
"q91_0: ??????M???????\n",
"         ?  ??????   \n",
"q91_1: ?????????M????\n",
"       ????? ? ??????\n",
"q91_2: ? X ????????M?\n",
"       ????? ?  ? ???\n",
"c24: 3/??????????????\n",
"             0  1  2 </pre>"
],
"text/plain": [
"            ???      \n",
"q91_0: ??????M???????\n",
"         ?  ??????   \n",
"q91_1: ?????????M????\n",
"       ????? ? ??????\n",
"q91_2: ? X ????????M?\n",
"       ????? ?  ? ???\n",
"c24: 3/??????????????\n",
"             0  1  2 "
]
},
"execution_count": 29,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"q = QuantumRegister(3)\n",
"c = ClassicalRegister(3)\n",
"qc = QuantumCircuit(q, c)\n",
"\n",
"\n",
"qc.ccx(q[0], q[1], q[2])\n",
"qc.measure(q, c)\n",
"qc.draw()\n"
]
},
{
```

```
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## No Cloning Theorem\n",
"\n",
"With multiple qubits of state $|\\psi \\rangle$, we can get an estimate of $\\alpha, \\beta$ but not with a single qubit because on measuring, it collapses to either one of its computational basis states $|0\\rangle$, or $|1\\rangle$\n",
"\n",
"Question is, can we clone a qubit to make mutliple copies of $|\\psi\\rangle$? (Need a circuit that takes a qubit and an ancillary qubit [Ancilla] as input and output two copies of $|\\psi\\rangle$) $\\rightarrow$ CNOT gate might be useful (due to the following copying action):\n",
"\n",
"$$CNOT|0\\rangle |0\\rangle \\rightarrow |0\\rangle |0\\rangle$$\n",
"$$CNOT|1\\rangle |0\\rangle \\rightarrow |1\\rangle |1\\rangle$$\n",
"\n",
"But, for any given arbitary qubit state,\n",
"$$CNOT(\\alpha |0\\rangle + \\beta|1\\rangle)\\rightarrow \\alpha |0\\rangle|0\\rangle+\\beta|1\\rangle|1\\rangle \\text{(Entanglement!)}$$\n",
"We wanted:\n",
"$$|\\psi\\rangle \\otimes |\\psi\\rangle = (\\alpha |0\\rangle + \\beta|1\\rangle)\\otimes(\\alpha |0\\rangle + \\beta|1\\rangle)$$\n",
"\n",
"Infact, no such circuit for copying can exist, i.e., no Unitary U exists for all unknown arbitrary states $|\\psi\\rangle$, $U|\\psi\\rangle|j\\rangle \\rightarrow |\\psi\\rangle|\\psi\\rangle$"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Code for creating Bell State $|\\phi^+\\rangle$"
]
},
{
"cell_type": "code",
"execution_count": 3,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre style=\"word-wrap: normal;white-space: pre;background: #fff0;line-height: 1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">      ?????     ???   \n",
"q0_0: ? H ???????M????\n",
"      ??????????????????\n",
"q0_1: ?????? X ?????M?\n",
"           ????? ? ???\n",
"c0: 2/??????????????????\n",
"                 0  1 </pre>"
```

```
],
"text/plain": [
"      ?????     ???   \n",
"q0_0: ? H ???????M????\n",
"      ??????????????\n",
"q0_1: ?????? X ?????M?\n",
"           ????? ? ???\n",
"c0: 2/??????????????\n",
"                 0  1 "
]
},
"execution_count": 3,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"q = QuantumRegister(2)\n",
"c = ClassicalRegister(2)\n",
"qc = QuantumCircuit(q, c)\n",
"\n",
"qc.h(q[0])\n",
"qc.cx(q[0], q[1])\n",
"qc.measure(q, c)\n",
"\n",
"qc.draw()"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"{'11': 513, '00': 511}\n"
]
}
],
"source": [
"backend = Aer.get_backend('qasm_simulator')\n",
"qjob = execute(qc, backend)\n",
"counts = qjob.result().get_counts()\n",
"print(counts)"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"### Code for creating Bell State $|GHZ_{+}\\rangle$"
```

```
]
},
{
"cell_type": "code",
"execution_count": 25,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre    style=\"word-wrap:    normal;white-space:    pre;background:    #fff0;line-height:
1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">       ?????          ???
   \n",
"q85_0: ? H ????????????M???????\n",
"       ??????????     ??????   \n",
"q85_1: ?????? X ??????????M????\n",
"            ?????????? ? ??????\n",
"q85_2: ?????????? X ????????M?\n",
"                 ????? ?  ? ???\n",
"c21: 3/?????????????????????????\n",
"                       0  1  2 </pre>"
],
"text/plain": [
"       ?????          ???      \n",
"q85_0: ? H ????????????M???????\n",
"       ??????????     ??????   \n",
"q85_1: ?????? X ??????????M????\n",
"            ?????????? ? ??????\n",
"q85_2: ?????????? X ????????M?\n",
"                 ????? ?  ? ???\n",
"c21: 3/?????????????????????????\n",
"                       0  1  2 "
]
},
"execution_count": 25,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"q = QuantumRegister(3)\n",
"c = ClassicalRegister(3)\n",
"qc = QuantumCircuit(q, c)\n",
"\n",
"qc.h(q[0])\n",
"qc.cx(q[0], q[1])\n",
"qc.cx(q[1], q[2])\n",
"qc.measure(q, c)\n",
"qc.draw()\n"
]
},
{
"cell_type": "code",
"execution_count": 26,
```

```
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"{'111': 527, '000': 497}\n"
]
}
],
"source": [
"backend = Aer.get_backend('qasm_simulator')\n",
"qjob = execute(qc, backend)\n",
"counts = qjob.result().get_counts()\n",
"print(counts)"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Multi-Qubit Measurement\n",
"\n",
"Probability of obtaining $i$ as the measurement outcome for a system in state $|\\psi\\rangle$ is defined using the measurement operator $M_i$ as:\n",
"$$Pr(i) = \\langle\\psi|M^{\\dagger}_i M_i |\\psi\\rangle$$\n",
"And on obtaining the outcome $i$, the ssytem collapses to:\n",
"$$|\\psi'\\rangle = \\frac{M_i|\\psi\\rangle}{\\sqrt{\\langle\\psi|M^{\\dagger}_i M_i |\\psi\\rangle}}$$ \n",
"(Post measurement state is normalized)\n",
"\n",
"For eg., Probability of obtaining outcome $00$ upon measuring $|\\psi\\rangle$ is:\n",
"$$Pr(00) = \\langle\\psi|(|00\\rangle\\langle 00|)^{\\dagger}(|00\\rangle\\langle 00|)|\\psi\\rangle$$\n",
"Let $|\\psi \\rangle = \\frac{1}{2}(|00\\rangle + |01\\rangle + |10\\rangle + |11\\rangle)$, so\n",
"\n",
"$$\\begin{align*}(|00\\rangle\\langle 00|)|\\psi\\rangle &= (|00\\rangle\\langle 00|)\\bigg[\\frac{1}{2}(|00\\rangle + |01\\rangle + |10\\rangle + |11\\rangle)\\bigg] \\\\\n",
"&=\\frac{1}{2}|00\\rangle \\bigg(\\langle 00|00\\rangle + \\langle 00|01\\rangle+\\langle 00|10\\rangle+\\langle 00|11\\rangle\\bigg) \\\\\n",
"&=\\frac{1}{2}|00\\rangle \\end{align*}$$\n",
"Now, \n",
"$$ \\langle\\psi|(|00\\rangle\\langle 00|)^{\\dagger} = \\bigg((|00\\rangle \\langle 00|)|\\psi\\rangle\\bigg)^{\\dagger}$$\n",
"$$\\implies Pr(00) = (\\frac{1}{2}\\langle 00|)(\\frac{1}{2}|00\\rangle) = \\frac{1}{4}\\langle 00 | 00\\rangle = \\frac{1}{4}$$\n",
"\n",
"On measuring just the of the first qubit of the state $|\\psi \\rangle$, the probability of obtaining outcome $0$ can be computed as $\\frac{1}{2}$, and post measurement, the state of the system on measuring the first qubit as $0$ can be given as \n",
```

```
"$$|\\psi_0\\rangle = \\frac{1}{\\sqrt2}(|00\\rangle + |01\\rangle)$$\n",
"\n",
"Set of measurement operators must satisfy $$\\sum_i M_i^{\\dagger}M_i = I \\\\\\text{(Completeness Relation)}$$\n",
"\n",
"Projective measurements: Set of operators $P_i = M_i^{\\dagger}M_i$ satisfying completeness relation $\\sum_iP_i=1$ as well as that the operators are orthogonal, so $P_iP_{i'}=P_i$ only when $i'=i$\n",
"\n",
"Example of projective measurement is the following set of operators:\n",
"$$\\bigg\\{|00\\rangle \\langle 00|, |01\\rangle \\langle 01|, |10\\rangle \\langle 10|, |11\\rangle \\langle 11|\\bigg\\}$$"
]
}
],
"metadata": {
"kernelspec": {
"display_name": "quantum",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.11.3"
},
"orig_nbformat": 4
},
"nbformat": 4,
"nbformat_minor": 2
}
```


File: 4_Quantum Teleportation.ipynb

```
{
"cells": [
{
"cell_type": "code",
"execution_count": 2,
"metadata": {},
"outputs": [],
"source": [
"from dotenv import load_dotenv\n",
"import os\n",
"from qiskit import *\n",
```

```
"from qiskit_ibm_provider import IBMProvider\n",
"from math import *\n",
"import qiskit\n",
"\n",
"load_dotenv()\n",
"IBM_KEY = os.getenv(\"API_KEY\")\n",
"provider = IBMProvider()"
]
},
{
"attachments": {},
"cell_type": "markdown",
"metadata": {},
"source": [
"## Quantum Teleportation\n",
"\n"
]
},
{
"cell_type": "code",
"execution_count": 4,
"metadata": {},
"outputs": [
{
"data": {
"text/html": [
"<pre   style=\"word-wrap:   normal;white-space:   pre;background:   #fff0;line-height:
1.1;font-family: &amp;quot;Courier New&amp;quot;,Courier,monospace\">        ?????
???????? ?                                  \n",
"q1_0: ? H ??????????? H ??M?????????????????????????????????????\n",
"      ?????     ????????????? ?                                      \n",
"q1_1: ? H ??????? X ???M?????????????????????????????????????\n",
"      ?????????????? ???  ?  ?    ?????       ?????    ????????\n",
"q1_2: ?????? X ?????????????????????? X ????????? Z ?????? H ??M?\n",
"           ?????       ?   ?  ?    ?????       ?????    ????????\n",
"                       ?   ?    ?????????????????????????      ? \n",
"c2: 2/?????????????????????????????? c2_1=0x1 ?? c2_0=0x1 ?????????\n",
"                       1   0    ?????????????????????????      ? \n",
"c3: 1/?????????????????????????????????????????????????????????\n",
"                                                                0 </pre>"
],
"text/plain": [
"      ?????          ???????? ?                                      \n",
"q1_0: ? H ??????????? H ??M?????????????????????????????????????\n",
"      ?????     ????????????? ?                                      \n",
"q1_1: ? H ??????? X ???M?????????????????????????????????????\n",
"      ?????????????? ???  ?  ?    ?????       ?????    ????????\n",
"q1_2: ?????? X ?????????????????????? X ????????? Z ?????? H ??M?\n",
"           ?????       ?   ?  ?    ?????       ?????    ????????\n",
"                       ?   ?    ?????????????????????????      ? \n",
"c2: 2/?????????????????????????????? c2_1=0x1 ?? c2_0=0x1 ?????????\n",
"                       1   0    ?????????????????????????      ? \n",
"c3: 1/?????????????????????????????????????????????????????????\n",
"                                                                0 "
```

```
]
},
"execution_count": 4,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
"q = QuantumRegister(3)\n",
"ca = ClassicalRegister(2)\n",
"cb = ClassicalRegister(1)\n",
"\n",
"qc = QuantumCircuit(q, ca, cb)\n",
"# Creating entaglement\n",
"qc.h(q[1])\n",
"qc.cx(q[1], q[2])\n",
"\n",
"# Creating a dummy state that is to be teleported\n",
"qc.h(q[0])\n",
"qc.cx(q[0], q[1])\n",
"qc.h(q[0])\n",
"qc.measure(q[0], ca[0])\n",
"qc.measure(q[1], ca[1])\n",
"qc.barrier()\n",
"\n",
"# Controlled on the o/p receieved by Bob, he performs X gate and Z gate\n",
"qc.x(q[2]).c_if(ca[1] ,1)\n",
"qc.z(q[2]).c_if(ca[0] ,1)\n",
"\n",
"# Check if state obtained is same as one to be teleported (H^{dag} = H), So output on
measuring q[2] should always be |0>\n",
"qc.h(q[2])\n",
"qc.measure(q[2], cb[0])\n",
"\n",
"qc.draw()"
]
},
{
"cell_type": "code",
"execution_count": 5,
"metadata": {},
"outputs": [
{
"name": "stdout",
"output_type": "stream",
"text": [
"{'0 11': 250, '0 00': 254, '0 01': 253, '0 10': 267}\n"
]
}
],
"source": [
"backend = Aer.get_backend('qasm_simulator')\n",
"qjob = execute(qc, backend)\n",
"counts = qjob.result().get_counts()\n",
```

```
"print(counts)\n",
"# Output structure is cb[0] ca[0]ca[1]\n",
"# Always get 0 for q[2] "
]
}
],
"metadata": {
"kernelspec": {
"display_name": "quantum",
"language": "python",
"name": "python3"
},
"language_info": {
"codemirror_mode": {
"name": "ipython",
"version": 3
},
"file_extension": ".py",
"mimetype": "text/x-python",
"name": "python",
"nbconvert_exporter": "python",
"pygments_lexer": "ipython3",
"version": "3.11.3"
},
"orig_nbformat": 4
},
"nbformat": 4,
"nbformat_minor": 2
}
```

Repository: SnapScribe

File: Backend_resources.md

```
# Backend Development Resources
This file contains a comprehensive list of resources that can be valuable for backend development in the SnapScribe project. These resources cover various aspects such as backend frameworks and API development.

## Backend Frameworks
For both of the suggested frameworks, basic familiarity with Python is necessary.
- [Flask](https://flask.palletsprojects.com/en/2.3.x/) - A lightweight and versatile Python web framework. Flask provides a simple yet powerful foundation for building web applications and APIs.

- [CS50 2020 lec 9 - Flask](https://youtu.be/x_c8pTW8ZUc) - Great tutorial for building a Flask web app. Holds your hand all the way through and explains all the basic concepts. Probably all the Flask you need to learn for this project. But remember that they use Jinja syntax which we won't be using or needing, it is old. We will need to connect it to a React frontend.

- [Django](https://docs.djangoproject.com/en/4.2/) - A high-level Python web framework known for its "batteries included" approach. It offers robust features for building scalable and secure web applications.

The preferred method for this project is a Flask web server serving the ML model, but you can use Django if you're more comfortable with that. The high level concept remains the same with any of the two, which is that you will have to build routes in the web app to serve predictions.

## HTTP Basics

- [CS50 2020 Lec 8 - Web](https://youtu.be/5g0x2xv3aHU) - Great introduction to HTTP protocols and type of requests like GET and POST, their differences and when each one is used. Also gives an intro as to what headers are. Great starting point.

- [HTTP MDN Docs](https://developer.mozilla.org/en-US/docs/Web/HTTP) - Very in-depth. Do not need to learn all of this. Don't even try to remember or read through this whole thing. Use as and when needed as reference material.

- [HTTP Headers MDN Docs](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) - Same as above, use as reference material.

## Sending Requests and Fetching Responses

- [Requests in Python](https://docs.python-requests.org/en/latest/) - Requests is an elegant and simple HTTP library for Python. You might need to use this.

For sending and fetching requests in React, you'll need the following (basically the following will communicate between the frontend and backend - user uploads image to the frontend, that is sent to the backend and the backend responds with a caption that is then displayed by the frontend).
```

- [Form Handling - Net Ninja](https://www.youtube.com/playlist?list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d) - Videos #14-32 are essential. Won't take long. Definitely watch them.

- [Axios](https://axios-http.com/docs/intro) - A popular JavaScript library for making HTTP requests from web browsers or Node.js. Axios is great for the handling of responses that it provides.

- [Axios freeCodeCamp Tutorial](https://www.freecodecamp.org/news/how-to-use-axios-with-react/) - Guide to get started with Axios and React.

- [Fetch API MDN Docs](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) - The MDN Web Docs guide to the Fetch API, a modern browser feature for making asynchronous HTTP requests.

- [Asynchronous JavaScript](https://www.freecodecamp.org/news/asynchronous-javascript) - Lean about Promises, Callbacks, and asynchronous code handling in JavaScript. You will be able to apply this in React after that.

API Resources
Learn about APIs, their types. Specifically the REST API design is very useful. Learn that by Googling. You will be building the API with Flask. But getting some theory in your head won't hurt, but don't spend much time on this.

[Postman](https://www.postman.com/) - Good tool for testing API routes with various types of input. You can learn how to use this for API testing by Googling and YouTube.

File: CODE_OF_CONDUCT.md

SnapScribe Code of Conduct for CSOC
Introduction
SnapScribe is committed to creating an open and inclusive community where everyone feels welcome, respected, and valued. We expect all community members, including contributors, maintainers, and users, to adhere to this Code of Conduct when participating in any SnapScribe-related activities or contributing to the project. This Code of Conduct outlines our shared values and the behavior we expect from individuals within our community.

Our Values
- Respect: Treat others with respect, kindness, and empathy. Be considerate of different perspectives and experiences.
- Inclusivity: Foster an inclusive and diverse environment where everyone feels safe and valued, regardless of their race, ethnicity, gender, sexual orientation, age, ability, or any other personal attribute.
- Collaboration: Encourage collaboration and constructive communication. Work together to achieve common goals and resolve conflicts in a respectful manner.
- Openness: Embrace openness, transparency, and accountability in our interactions and decision-making processes.
- Professionalism: Maintain a professional and courteous demeanor. Refrain from engaging in any form of harassment, discrimination, or disruptive behavior.
Expected Behavior
All community members are expected to:

- Be respectful and considerate of others' opinions, ideas, and experiences.
- Use inclusive language and avoid any offensive, discriminatory, or derogatory remarks or behavior.
- Be open to constructive feedback and provide feedback in a respectful and constructive manner.
- Exercise empathy and assume good intentions in all interactions.
- Be mindful of the impact of your words and actions on others in the community.
- Respect the privacy and confidentiality of others.
- Be accountable for your actions and their consequences.

Unacceptable Behavior

The following behaviors are considered unacceptable within the SnapScribe community:

- Harassment, bullying, or intimidation in any form.
- Discrimination or offensive comments related to race, ethnicity, gender, sexual orientation, age, ability, or any other personal attribute.
- Use of derogatory, offensive, or inappropriate language or imagery.
- Personal attacks, insults, or trolling.
- Disruptive or disrespectful behavior during community discussions or events.
- Any other behavior that violates the principles outlined in this Code of Conduct.

File: CONTRIBUTE.md

Contributing to SnapScribe

Thank you for your interest in contributing to SnapScribe! We appreciate your support in making this project even better. Before contributing, take a look at the following guidelines, which will help you get started with making your first PRs and resolving issues!

How to Contribute

1. Fork the repository to your GitHub account.
2. Create a new branch from the main branch for your contributions.
3. Make your changes and improvements in the branch you created.
4. Ensure your code adheres to the project's coding style and follows best practices.
5. Test your changes locally to ensure they function as expected.
6. Commit your changes with descriptive commit messages.
7. Push your branch to your forked repository.
8. Open a pull request (PR) from your branch to the main branch of the main repository.
9. Provide a clear and detailed description of the changes you made in the PR.
10. Be responsive to any feedback or comments provided during the review process.

Making Your First Pull Request (PR)

1. Find an issue labeled as "good first issue" among list of issues. These are great starting points.

2. Comment on the issue to let others know that you're interested in working on it. This helps prevent duplicate efforts and allows for coordination if multiple contributors are interested in the same issue.

3. Fork the repository to your GitHub account by clicking the "Fork" button on the top-right corner of the repository page.

4. Clone your forked repository to your local machine using the following command:

`git clone https://github.com/your-username/SnapScribe.git`

5. Create a new branch for your changes using a descriptive branch name:

`git checkout -b my-first-contribution`

6. Make the necessary changes and improvements to the codebase.

7. Test your changes locally to ensure they work as expected.

8. Commit your changes with a clear and concise commit message:
`git commit -m "Add feature XYZ"`

9. Push your changes to your forked repository:

`git push origin my-first-contribution`

10. Visit the original repository on GitHub and click the "New pull request" button.

11. Provide a descriptive title and detailed description for your pull request, explaining the changes you made.

12. Submit the pull request and wait for the project maintainers to review your changes. Be patient and responsive to any feedback or suggestions provided.

Code Style and Guidelines
- Follow the coding style and conventions used in the existing codebase.
- Write clear and concise code with meaningful variable and function names.
- Include comments in your code where necessary to enhance readability.
- Ensure your code is well-documented, including inline comments and docstrings.
- Write unit tests for new features or modifications to existing functionality.
- Run existing tests to ensure you haven't introduced any regressions.
Reporting Issues
- If you encounter any bugs, issues, or have suggestions for improvements, please submit an issue on the GitHub repository. When submitting an issue, please include the following information:

1. A clear and descriptive title.
2. A detailed description of the issue or suggestion.
3. Steps to reproduce the issue, if applicable.
4. Any relevant logs, error messages, or screenshots.

Documentation
Improvements to the project's documentation are highly valuable. If you find any gaps or areas that need clarification, please feel free to submit documentation updates or create new documentation where needed.

Collaboration and Communication
We encourage collaboration and open communication among contributors. If you have any questions or need assistance, please use the CSOC Discord Server's relevant channel. You can also reach out through GitHub discussions or comments on the relevant PR or issue.

File: Deployment_resources.md

Deployment Resources
This file provides a collection of resources for deploying the SnapScribe web application. Below are various options and guides for deploying your project on different platforms.

GitHub Pages
- [GitHub Pages](https://pages.github.com/) - Official documentation for deploying static websites directly from your GitHub repository to GitHub Pages. It provides step-by-step instructions to set up and configure your deployment.

- [Deploying a React App to GitHub Pages](https://create-react-app.dev/docs/deployment/#github-pages) - A guide that demonstrates how to deploy a React application to GitHub Pages using the gh-pages package and the npm run deploy command.

Netlify
- [Netlify](https://docs.netlify.com/) - A popular platform for deploying static websites, providing features like continuous deployment, custom domain support, and built-in CDN. Their official documentation provides detailed instructions on setting up and deploying your project.

- [Deploying a React App to Netlify](https://create-react-app.dev/docs/deployment/#netlify) - A step-by-step guide on deploying a React application to Netlify. It covers configuring build settings, setting up environment variables, and connecting your repository.

PythonAnywhere
- [PythonAnywhere](https://www.pythonanywhere.com/) - A cloud-based Python hosting service that allows you to deploy and run web applications. Their official documentation provides instructions for deploying Python web apps and configuring the necessary settings.

- [Deploying a Flask App on PythonAnywhere](https://help.pythonanywhere.com/pages/Flask/) - A detailed tutorial on deploying a Flask application on PythonAnywhere. It covers setting up the virtual environment, configuring the web server, and deploying the application.

File: Frontend_resources.md

Frontend Development Resources
The frontend for SnapScribe is meant to be a React.js web app styles with Tailwind CSS. For this part of the project, the most useful resource will probably be the documentations of the respective frameworks, along with a few YouTube guides here and there. The Art of Googling is *ESSENTIAL* for this part of the project specially. StackOverflow, YouTube, Reddit are your best friends.

First you will have to come up with a rough overview of a design for the website. What

kind of components are necessary, what pages will contain what, although a brief guideline for that will be provided soon. Styling will be totally upto the contributors.

Design and UI Inspiration
- [Dribbble](https://dribbble.com/) - An online community of designers showcasing their work. Explore the platform for UI/UX inspiration, trends, and design ideas.
- [Behance](https://www.behance.net/) - Discover creative projects and portfolios from designers around the world. Explore different design styles and layouts.

Frontend Frameworks and Libraries
- [React.js](https://react.dev/learn) - A popular JavaScript library for building user interfaces. It offers a component-based approach and efficient rendering for creating interactive UIs.

- [Tailwind CSS](https://tailwindcss.com/docs/installation) - A utility-first CSS framework that allows you to rapidly build custom user interfaces. It provides a set of pre-defined CSS classes to streamline styling.

- [React Router](https://reactrouter.com/en/main) - For creating multi-page web apps in React.

Tutorials, YouTube Playlists
- FreeCodeCamp - An online learning platform with interactive tutorials and projects for frontend development. It covers HTML, CSS, JavaScript, and popular frontend frameworks. Look at their website as well as YouTube channel for guided tutorials

- Codecademy - Good place for learning Javascript, JSX and start with React.js. Before learning React.js, you must be comfortable with the basics of Javascript and JSX.

- [Javascript Mastery](https://www.youtube.com/@javascriptmastery) - Great guided projects and will get you up and running with React very quick! Highly recommended. They describe a great project structure at the beginning of every video which would help to keep modularity of React.

- [Net Ninja](https://www.youtube.com/@NetNinja) - Great playlists on more technical aspects of React like Form handling, submission handling, etc.

- [React Tutorial by Net Ninja](https://www.youtube.com/playlist?list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d) - The definitive playlist for Form handling and explanation of a lot of parts in bite sized quick videos.

- [React Router Example](https://youtu.be/xMNhDf5-hvk) - Example of how to create a multi page web app if you need it.

UI Component Libraries
- [Material-UI](https://mui.com/material-ui/getting-started/overview/) - A popular React component library that follows Google's Material Design guidelines. It provides a wide range of pre-built UI components and styles.

- [Chakra UI](https://chakra-ui.com/getting-started) - A simple and customizable UI component library for React. It focuses on accessibility, developer experience, and easy theming.