```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": [],
   "collapsed_sections": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "markdown",
   "source": [
    "Import the Dataset"
   ],
   "metadata": {
    "id": "ZIGeX2SN4end"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "from google.colab import files\n",
    "uploaded = files.upload()"
    ],
    "metadata": {
     "id": "9QqLNbfz4ns0",
     "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 73
     },
     "outputId": "40807474-8aa2-4921-dfcf-c7bfc742dbbe"
    },
    "execution_count": 2,
    "outputs": [
     {
      "output_type": "display_data",
      "data": {
       "text/plain": [
        "<IPython.core.display.HTML object>"
       ],
       "text/html": [
        "\n",
        "     <input type=\"file\" id=\"files-f0261ce0-8d3d-4f57-84e7-fffbdeca1ba0\" name=\"files[]\"
multiple disabled\n",
        "        style=\"border:none\" />\n",
        "     <output id=\"result-f0261ce0-8d3d-4f57-84e7-fffbdeca1ba0\">\n",
        "      Upload widget is only available when the cell has been executed in the\n",
        "      current browser session. Please rerun this cell to enable.\n",
        "      </output>\n",
        "      <script>// Copyright 2017 Google LLC\n",
        "//\n",
        "// Licensed under the Apache License, Version 2.0 (the \"License\");\n",
        "// you may not use this file except in compliance with the License.\n",
        "// You may obtain a copy of the License at\n",
"//\n",
"//      http://www.apache.org/licenses/LICENSE-2.0\n",
"//\n",
"// Unless required by applicable law or agreed to in writing, software\n",
"// distributed under the License is distributed on an \"AS IS\" BASIS,\n",
"// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.\n",
"// See the License for the specific language governing permissions and\n",
"// limitations under the License.\n",
"\n",
"/**\n",
" * @fileoverview Helpers for google.colab Python module.\n",
" */\n",
"(function(scope) {\n",
"function span(text, styleAttributes = {}) {\n",
"  const element = document.createElement('span');\n",
"  element.textContent = text;\n",
"  for (const key of Object.keys(styleAttributes)) {\n",
"    element.style[key] = styleAttributes[key];\n",
"  }\n",
"  return element;\n",
"}\n",
"\n",
"// Max number of bytes which will be uploaded at a time.\n",
"const MAX_PAYLOAD_SIZE = 100 * 1024;\n",
"\n",
"function _uploadFiles(inputId, outputId) {\n",
"  const steps = uploadFilesStep(inputId, outputId);\n",
"  const outputElement = document.getElementById(outputId);\n",
"  // Cache steps on the outputElement to make it available for the next call\n",
"  // to uploadFilesContinue from Python.\n",
"  outputElement.steps = steps;\n",
"\n",
"  return _uploadFilesContinue(outputId);\n",
"}\n",
"\n",
"// This is roughly an async generator (not supported in the browser yet),\n",
"// where there are multiple asynchronous steps and the Python side is going\n",
"// to poll for completion of each step.\n",
"// This uses a Promise to block the python side on completion of each step,\n",
"// then passes the result of the previous step as the input to the next step.\n",
"function _uploadFilesContinue(outputId) {\n",
"  const outputElement = document.getElementById(outputId);\n",
"  const steps = outputElement.steps;\n",
"\n",
"  const next = steps.next(outputElement.lastPromiseValue);\n",
"  return Promise.resolve(next.value.promise).then((value) => {\n",
"    // Cache the last promise value to make it available to the next\n",
"    // step of the generator.\n",
"    outputElement.lastPromiseValue = value;\n",
"    return next.value.response;\n",
"  });\n",
"}\n",
"\n",
"/**\n",
" * Generator function which is called between each async step of the upload\n",
" * process.\n",
" * @param {string} inputId Element ID of the input file picker element.\n",
" * @param {string} outputId Element ID of the output display.\n",
" * @return {!Iterable<!Object>} Iterable of next steps.\n",
" */\n",
"function* uploadFilesStep(inputId, outputId) {\n",
"  const inputElement = document.getElementById(inputId);\n",
"  inputElement.disabled = false;\n",
"\n",
"  const outputElement = document.getElementById(outputId);\n",
"  outputElement.innerHTML = '';\n",
"\n",
"  const pickedPromise = new Promise((resolve) => {\n",
"    inputElement.addEventListener('change', (e) => {\n",
"      resolve(e.target.files);\n",
"    });\n",
"  });\n",
"\n",
"  const cancel = document.createElement('button');\n",
"  inputElement.parentElement.appendChild(cancel);\n",
"  cancel.textContent = 'Cancel upload';\n",
"  const cancelPromise = new Promise((resolve) => {\n",
"    cancel.onclick = () => {\n",
"      resolve(null);\n",
"    };\n",
"  });\n",
"\n",
"  // Wait for the user to pick the files.\n",
"  const files = yield {\n",
"    promise: Promise.race([pickedPromise, cancelPromise]),\n",
"    response: {\n",
"      action: 'starting',\n",
"    }\n",
"  };\n",
"\n",
"  cancel.remove();\n",
"\n",
"  // Disable the input element since further picks are not allowed.\n",
"  inputElement.disabled = true;\n",
"\n",
"  if (!files) {\n",
"    return {\n",
"      response: {\n",
"        action: 'complete',\n",
"      }\n",
"    };\n",
"  }\n",
"\n",
"  for (const file of files) {\n",
"    const li = document.createElement('li');\n",
"    li.append(span(file.name, {fontWeight: 'bold'}));\n",
"    li.append(span(\n",
"        `(${file.type || 'n/a'}) - ${file.size} bytes, ` +\n",
"        `last modified: ${\n",
"            file.lastModifiedDate ? file.lastModifiedDate.toLocaleDateString() :\n",
"                                    'n/a'} - `));\n",
"    const percent = span('0% done');\n",
"    li.appendChild(percent);\n",
"\n",
"    outputElement.appendChild(li);\n",
"\n",
"    const fileDataPromise = new Promise((resolve) => {\n",
"      const reader = new FileReader();\n",
"      reader.onload = (e) => {\n",
"        resolve(e.target.result);\n",
"      };\n",
"      reader.readAsArrayBuffer(file);\n",
"    });\n",
"    // Wait for the data to be ready.\n",
"    let fileData = yield {\n",
"      promise: fileDataPromise,\n",
"      response: {\n",
"        action: 'continue',\n",
"      }\n",
"    };\n",
"\n",
"    // Use a chunked sending to avoid message size limits. See b/62115660.\n",
"    let position = 0;\n",
"    do {\n",
"      const length = Math.min(fileData.byteLength - position, MAX_PAYLOAD_SIZE);\n",
"      const chunk = new Uint8Array(fileData, position, length);\n",
"      position += length;\n",
"\n",
"      const base64 = btoa(String.fromCharCode.apply(null, chunk));\n",
"      yield {\n",
"        response: {\n",
"          action: 'append',\n",
"          file: file.name,\n",
"          data: base64,\n",
"        },\n",
"      };\n",
"\n",
"      let percentDone = fileData.byteLength === 0 ?\n",
"          100 :\n",
"          Math.round((position / fileData.byteLength) * 100);\n",
"      percent.textContent = `${percentDone}% done`;\n",
"\n",
"    } while (position < fileData.byteLength);\n",
"  }\n",
"\n",
        "  // All done.\n",
        "  yield {\n",
        "    response: {\n",
        "      action: 'complete',\n",
        "    }\n",
        "  };\n",
        "}\n",
        "\n",
        "scope.google = scope.google || {};\n",
        "scope.google.colab = scope.google.colab || {};\n",
        "scope.google.colab._files = {\n",
        "  _uploadFiles,\n",
        "  _uploadFilesContinue,\n",
        "};\n",
        "})(self);\n",
        "</script> "
       ]
      },
      "metadata": {}
     },
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "Saving spam.csv to spam.csv\n"
      ]
     }
    ]
   },
   {
    "cell_type": "markdown",
  "source": [
   "**Import required libraries**\n",
   "\n"
  ],
  "metadata": {
   "id": "ghP1qRWK0O12"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "import csv\n",
   "import tensorflow as tf\n",
   "import pandas as pd\n",
   "import numpy as np\n",
   "import matplotlib.pyplot as plt\n",
   "from tensorflow.keras.preprocessing.text import Tokenizer\n",
   "from tensorflow.keras.preprocessing.sequence import pad_sequences\n",
   "import nltk\n",
   "nltk.download('stopwords')  \n",
   "from nltk.corpus import stopwords\n",
   "STOPWORDS = set(stopwords.words('english'))"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "ulfQWre5lwkO",
   "outputId": "3d132bbd-a86a-4911-ff13-9856f6f5c28b"
  },
  "execution_count": 5,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stderr",
      "text": [
        "[nltk_data] Downloading package stopwords to /root/nltk_data...\n",
        "[nltk_data]   Unzipping corpora/stopwords.zip.\n"
      ]
    }
  ]
},
{
  "cell_type": "markdown",
  "source": [
    "**Import dataset**"
  ],
  "metadata": {
    "id": "dbpTZRgo0aUT"
  }
},
{
  "cell_type": "code",
  "source": [
    "import io\n",
    "dataset = pd.read_csv(io.BytesIO(uploaded['spam.csv']), encoding = \"ISO-8859-1\")"
  ],
  "metadata": {
    "id": "NJSQAU8bmJVt"
  },
  "execution_count": 6,
  "outputs": []
```

```
},
{
 "cell_type": "code",
 "source": [
  "dataset"
 ],
 "metadata": {
  "id": "Zfdffoymnj3B",
  "outputId": "feaf46c3-adec-4ecd-90e4-54bfaaa2d8f8",
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 424
  }
 },
 "execution_count": 7,
 "outputs": [
  {
   "output_type": "execute_result",
   "data": {
    "text/plain": [
     "       v1                                                 v2 Unnamed: 2  \\\n",
     "0     ham  Go until jurong point, crazy.. Available only ...        NaN   \n",
     "1     ham                      Ok lar... Joking wif u oni...        NaN   \n",
     "2    spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN   \n",
     "3     ham  U dun say so early hor... U c already then say...        NaN   \n",
     "4     ham  Nah I don't think he goes to usf, he lives aro...        NaN   \n",
     "...   ...                                                ...        ...   \n",
     "5567  spam  This is the 2nd time we have tried 2 contact u...        NaN   \n",
     "5568   ham              Will Ì_ b going to esplanade fr home?        NaN   \n",
     "5569   ham  Pity, * was in mood for that. So...any other s...        NaN   \n",
     "5570   ham  The guy did some bitching but I acted like i'd...        NaN   \n",
  "5571   ham                     Rofl. Its true to its name       NaN   \n",
  "\n",
  "    Unnamed: 3 Unnamed: 4  \n",
  "0          NaN        NaN  \n",
  "1          NaN        NaN  \n",
  "2          NaN        NaN  \n",
  "3          NaN        NaN  \n",
  "4          NaN        NaN  \n",
  "...        ...        ...  \n",
  "5567       NaN        NaN  \n",
  "5568       NaN        NaN  \n",
  "5569       NaN        NaN  \n",
  "5570       NaN        NaN  \n",
  "5571       NaN        NaN  \n",
  "\n",
  "[5572 rows x 5 columns]"
 ],
 "text/html": [
  "\n",
  "  <div id=\"df-4418c03a-8e1c-4e9c-ac20-1c8ea67dce4e\">\n",
  "    <div class=\"colab-df-container\">\n",
  "      <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>v1</th>\n",
"      <th>v2</th>\n",
"      <th>Unnamed: 2</th>\n",
"      <th>Unnamed: 3</th>\n",
"      <th>Unnamed: 4</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>ham</td>\n",
"      <td>Go until jurong point, crazy.. Available only ...</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>ham</td>\n",
"      <td>Ok lar... Joking wif u oni...</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>spam</td>\n",
"      <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>ham</td>\n",
"      <td>U dun say so early hor... U c already then say...</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>ham</td>\n",
"      <td>Nah I don't think he goes to usf, he lives aro...</td>\n",
"      <td>NaN</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",
"   </tr>\n",
"   <tr>\n",
"     <th>5567</th>\n",
"     <td>spam</td>\n",
"     <td>This is the 2nd time we have tried 2 contact u...</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>5568</th>\n",
"     <td>ham</td>\n",
"     <td>Will Ì_ b going to esplanade fr home?</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>5569</th>\n",
"     <td>ham</td>\n",
"     <td>Pity, * was in mood for that. So...any other s...</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>5570</th>\n",
"     <td>ham</td>\n",
"     <td>The guy did some bitching but I acted like i'd...</td>\n",
"     <td>NaN</td>\n",
```

"   <td>NaN</td>\n",
"   <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"   <th>5571</th>\n",
"   <td>ham</td>\n",
"   <td>Rofl. Its true to its name</td>\n",
"   <td>NaN</td>\n",
"   <td>NaN</td>\n",
"   <td>NaN</td>\n",
"   </tr>\n",
" </tbody>\n",
"</table>\n",
"<p>5572 rows × 5 columns</p>\n",
"</div>\n",
"   <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-4418c03a-8e1c-4e9c-ac20-1c8ea67dce4e')\"\n",
"   title=\"Convert this dataframe to an interactive table.\"\n",
"   style=\"display:none;\">\n",
"   \n",
" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",
"   width=\"24px\">\n",
"   <path d=\"M0 0h24v24H0V0z\" fill=\"none\"/>\n",
"   <path d=\"M18.56 5.44l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94zm-11 1L8.5 8.5l.94-2.06 2.06-.94-2.06-.94L8.5 2.5l-.94 2.06-2.06.94zm10 10l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94z\"/><path d=\"M17.41 7.96l-1.37-1.37c-.4-.4-.92-.59-1.43-.59-.52 0-1.04.2-1.43.59L10.3 9.45l-7.72 7.72c-.78.78-.78 2.05 0 2.83L4 21.41c.39.39.9.59 1.41.59.51 0 1.02-.2 1.41-.59l7.78-7.78 2.81-2.81c.8-.78.8-2.07 0-2.86zM5.41 20L4 18.59l7.72-7.72 1.47 1.35L5.41 20z\"/>\n",
" </svg>\n",
"   </button>\n",
"   \n",
" <style>\n",

```
"    .colab-df-container {\n",
"      display:flex;\n",
"      flex-wrap:wrap;\n",
"      gap: 12px;\n",
"    }\n",
"\n",
"    .colab-df-convert {\n",
"      background-color: #E8F0FE;\n",
"      border: none;\n",
"      border-radius: 50%;\n",
"      cursor: pointer;\n",
"      display: none;\n",
"      fill: #1967D2;\n",
"      height: 32px;\n",
"      padding: 0 0 0 0;\n",
"      width: 32px;\n",
"    }\n",
"\n",
"    .colab-df-convert:hover {\n",
"      background-color: #E2EBFA;\n",
"      box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67,
0.15);\n",
"      fill: #174EA6;\n",
"    }\n",
"\n",
"    [theme=dark] .colab-df-convert {\n",
"      background-color: #3B4455;\n",
"      fill: #D2E3FC;\n",
"    }\n",
"\n",
"    [theme=dark] .colab-df-convert:hover {\n",
```

"   background-color: #434B5C;\n",

"   box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

"   filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

"   fill: #FFFFFF;\n",

"   }\n",

" </style>\n",

"\n",

"   <script>\n",

"   const buttonEl =\n",

"   document.querySelector('#df-4418c03a-8e1c-4e9c-ac20-1c8ea67dce4e button.colab-df-convert');\n",

"   buttonEl.style.display =\n",

"   google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

"   async function convertToInteractive(key) {\n",

"   const element = document.querySelector('#df-4418c03a-8e1c-4e9c-ac20-1c8ea67dce4e');\n",

"   const dataTable =\n",

"   await google.colab.kernel.invokeFunction('convertToInteractive',\n",

"   [key], {});\n",

"   if (!dataTable) return;\n",

"\n",

"   const docLinkHtml = 'Like what you see? Visit the ' +\n",

"   '<a target=\"_blank\" href=https://colab.research.google.com/notebooks/data_table.ipynb>data table notebook</a>'\n",

"   + ' to learn more about interactive tables.';\n",

"   element.innerHTML = '';\n",

"   dataTable['output_type'] = 'display_data';\n",

"   await google.colab.output.renderOutput(dataTable, element);\n",

"   const docLink = document.createElement('div');\n",

"   docLink.innerHTML = docLinkHtml;\n",

"   element.appendChild(docLink);\n",

```
        "        }\n",
        "      </script>\n",
        "    </div>\n",
        "  </div>\n",
        "  "
       ]
      },
      "metadata": {},
      "execution_count": 7
     }
    ]
   },
   {
    "cell_type": "code",
    "source": [
     "vocab_size = 5000\n",
     "embedding_dim = 64\n",
     "max_length = 200\n",
     "trunc_type = 'post'\n",
     "padding_type = 'post'\n",
     "oov_tok = '<OOV>'\n",
     "training_portion = .8"
    ],
    "metadata": {
     "id": "bAikw7ObzQmM"
    },
    "execution_count": 8,
    "outputs": []
   },
   {
    "cell_type": "markdown",
  "source": [
   "Read the dataset and do pre-processing.\n",
   "\n",
   "To remove the stop words."
  ],
  "metadata": {
   "id": "E52Z7wbo1Bvl"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "articles = []\n",
   "labels = []\n",
   "\n",
   "with open(\"spam.csv\", 'r', encoding = \"ISO-8859-1\") as dataset:\n",
   "    reader = csv.reader(dataset, delimiter=',')\n",
   "    next(reader)\n",
   "    for row in reader:\n",
   "        labels.append(row[0])\n",
   "        article = row[1]\n",
   "        for word in STOPWORDS:\n",
   "            token = ' ' + word + ' '\n",
   "            article = article.replace(token, ' ')\n",
   "            article = article.replace(' ', ' ')\n",
   "        articles.append(article)\n",
   "print(len(labels))\n",
   "print(len(articles))"
  ],
  "metadata": {
   "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "tAdCQlDazXEB",
     "outputId": "4128c8c6-aa52-4ac1-9cf3-ce1883290cb9"
    },
    "execution_count": 10,
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "5572\n",
       "5572\n"
      ]
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "**Train the model**"
    ],
    "metadata": {
     "id": "sjdXzrlJ1Gw7"
    }
   },
   {
    "cell_type": "code",
    "source": [
     "train_size = int(len(articles) * training_portion)\n",
     "\n",
  "train_articles = articles[0: train_size]\n",
  "train_labels = labels[0: train_size]\n",
  "\n",
  "validation_articles = articles[train_size:]\n",
  "validation_labels = labels[train_size:]\n",
  "\n",
  "print(train_size)\n",
  "print(len(train_articles))\n",
  "print(len(train_labels))\n",
  "print(len(validation_articles))\n",
  "print(len(validation_labels))"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "XX7X4IQBzc9y",
  "outputId": "2bdb788e-0022-4c78-ec1a-0b6a9595f0ee"
 },
 "execution_count": 11,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "4457\n",
    "4457\n",
    "4457\n",
    "1115\n",
    "1115\n"
   ]
```

```
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)\n",
    "tokenizer.fit_on_texts(train_articles)\n",
    "word_index = tokenizer.word_index\n",
    "dict(list(word_index.items())[0:10])"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "IUUzkhjcze23",
    "outputId": "1adb2a2e-0baa-4254-9d21-6ebdafc609ba"
   },
   "execution_count": 12,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "{'<OOV>': 1,\n",
       " 'i': 2,\n",
       " 'u': 3,\n",
       " 'call': 4,\n",
       " 'you': 5,\n",
       " '2': 6,\n",
       " 'get': 7,\n",
        " \"i'm\": 8,\n",
        " 'ur': 9,\n",
        " 'now': 10}"
       ]
      },
      "metadata": {},
      "execution_count": 12
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "**Traning data to Sequences**"
    ],
    "metadata": {
     "id": "dKFspDee1Mpv"
    }
   },
   {
    "cell_type": "code",
    "source": [
     "train_sequences = tokenizer.texts_to_sequences(train_articles)\n",
     "print(train_sequences[10])"
    ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "_Y7dprX-ziM7",
     "outputId": "d6c316a8-304a-4ec0-a64e-3f8cd0db2124"
    },
    "execution_count": 13,
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "[8, 189, 37, 201, 30, 260, 293, 991, 222, 53, 153, 3815, 423, 46]\n"
      ]
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "**Train neural network for NLP**"
    ],
    "metadata": {
     "id": "YQCyIW2-1R94"
    }
   },
   {
    "cell_type": "code",
    "source": [
     "train_padded = pad_sequences(train_sequences, maxlen=max_length, padding=padding_type,
truncating=trunc_type)\n",
     "print(len(train_sequences[0]))\n",
     "print(len(train_padded[0]))\n",
     "\n",
     "print(len(train_sequences[1]))\n",
     "print(len(train_padded[1]))\n",
    "\n",
    "print(len(train_sequences[10]))\n",
    "print(len(train_padded[10]))"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "4TQjlc67zmuw",
    "outputId": "15a7398a-af56-421e-9860-3aa8e3a4a39f"
   },
   "execution_count": 14,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "16\n",
      "200\n",
      "6\n",
      "200\n",
      "14\n",
      "200\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "print(train_padded[10])"
```

```
],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "kumXYmkvzppD",
  "outputId": "a81246b1-103a-43cf-9c63-31bf3acb6ce2"
},
"execution_count": 15,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "[   8  189   37  201   30  260  293  991  222   53  153 3815  423   46\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0    0    0    0    0    0    0    0    0    0    0\n",
      "    0    0    0    0]\n"
    ]
  }
```

```
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "validation_sequences = tokenizer.texts_to_sequences(validation_articles)\n",
    "validation_padded = pad_sequences(validation_sequences, maxlen=max_length,
padding=padding_type, truncating=trunc_type)\n",
    "\n",
    "print(len(validation_sequences))\n",
    "print(validation_padded.shape)\n"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "UmYgpsHFzrFy",
    "outputId": "44be52a2-6988-4a48-8f44-e07457725266"
   },
   "execution_count": 16,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "1115\n",
      "(1115, 200)\n"
     ]
    }
   ]
  },
{
 "cell_type": "code",
 "source": [
  "label_tokenizer = Tokenizer()\n",
  "label_tokenizer.fit_on_texts(labels)\n",
  "\n",
  "training_label_seq = np.array(label_tokenizer.texts_to_sequences(train_labels))\n",
  "validation_label_seq = np.array(label_tokenizer.texts_to_sequences(validation_labels))\n",
  "print(training_label_seq[0])\n",
  "print(training_label_seq[1])\n",
  "print(training_label_seq[2])\n",
  "print(training_label_seq.shape)\n",
  "\n",
  "print(validation_label_seq[0])\n",
  "print(validation_label_seq[1])\n",
  "print(validation_label_seq[2])\n",
  "print(validation_label_seq.shape)"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "1OVkYRAmzvNA",
  "outputId": "57e602ae-d245-4ac0-dc61-c90eb3e13d0f"
 },
 "execution_count": 17,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
      "[1]\n",
      "[1]\n",
      "[2]\n",
      "(4457, 1)\n",
      "[1]\n",
      "[2]\n",
      "[1]\n",
      "(1115, 1)\n"
     ]
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])\n",
    "\n",
    "def decode_article(text):\n",
    "    return ' '.join([reverse_word_index.get(i, '?') for i in text])\n",
    "print(decode_article(train_padded[10]))\n",
    "print('---')\n",
    "print(train_articles[10])"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "ul0DBJ2Hzxq6",
    "outputId": "10da68ed-31a3-4c20-c43d-c52fbd02ef02"
   },
   "execution_count": 18,
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "i'm gonna home soon want talk stuff anymore tonight k i've cried enough today ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
?\n",
       "---\n",
       "I'm gonna home soon want talk stuff anymore tonight, k? I've cried enough today.\n"
      ]
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "**To implement LSTM**"
    ],
    "metadata": {
     "id": "2G-Vq1CN1dqj"
    }
   },
   {
    "cell_type": "code",
    "source": [
     "model = tf.keras.Sequential([\n",
     "  \n",
     "    tf.keras.layers.Embedding(vocab_size, embedding_dim),\n",
     "    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding_dim)),\n",
  "    tf.keras.layers.Dense(embedding_dim, activation='relu'),\n",
  "    tf.keras.layers.Dense(6, activation='softmax')\n",
  "])\n",
  "model.summary()"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "DHzP97zxz5U4",
  "outputId": "4de6a91c-cbb2-49bb-f83d-f8daccaff111"
 },
 "execution_count": 19,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Model: \"sequential\"\n",
    "_________________________________________________________________\n",
    " Layer (type)                Output Shape              Param #   \n",
    "=================================================================\n",
    " embedding (Embedding)       (None, None, 64)          320000    \n",
    "                                                                 \n",
    " bidirectional (Bidirectiona  (None, 128)              66048     \n",
    " l)                                                              \n",
    "                                                                 \n",
    " dense (Dense)               (None, 64)                8256      \n",
    "                                                                 \n",
    " dense_1 (Dense)             (None, 6)                 390       \n",
    "                                                                 \n",
        "=================================================================\n",
        "Total params: 394,694\n",
        "Trainable params: 394,694\n",
        "Non-trainable params: 0\n",
        "_________________________________________________________________\n"
      ]
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "print(set(labels))"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "jqQB_Yv_z9WQ",
    "outputId": "c1202dbe-b89f-43b8-f498-9f1966acda1e"
  },
  "execution_count": 20,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "{'ham', 'spam'}\n"
      ]
    }
  ]
  },
  {
   "cell_type": "code",
   "source": [
    "model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])\n",
    "num_epochs = 10\n",
    "history = model.fit(train_padded, training_label_seq, epochs=num_epochs,
validation_data=(validation_padded, validation_label_seq), verbose=2)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "6QqolQyIz_re",
    "outputId": "3a04d75f-0584-487b-f6c2-0e499b00d3ed"
   },
   "execution_count": 21,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Epoch 1/10\n",
      "140/140 - 33s - loss: 0.3079 - accuracy: 0.9237 - val_loss: 0.0536 - val_accuracy: 0.9874 -
33s/epoch - 239ms/step\n",
      "Epoch 2/10\n",
      "140/140 - 30s - loss: 0.0297 - accuracy: 0.9924 - val_loss: 0.0432 - val_accuracy: 0.9874 -
30s/epoch - 213ms/step\n",
      "Epoch 3/10\n",
      "140/140 - 31s - loss: 0.0129 - accuracy: 0.9973 - val_loss: 0.0366 - val_accuracy: 0.9901 -
31s/epoch - 220ms/step\n",
      "Epoch 4/10\n",
        "140/140 - 31s - loss: 0.0050 - accuracy: 0.9991 - val_loss: 0.0663 - val_accuracy: 0.9821 -
31s/epoch - 224ms/step\n",
        "Epoch 5/10\n",
        "140/140 - 28s - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.0467 - val_accuracy: 0.9892 -
28s/epoch - 203ms/step\n",
        "Epoch 6/10\n",
        "140/140 - 29s - loss: 0.0021 - accuracy: 0.9996 - val_loss: 0.0495 - val_accuracy: 0.9874 -
29s/epoch - 206ms/step\n",
        "Epoch 7/10\n",
        "140/140 - 30s - loss: 0.0012 - accuracy: 0.9998 - val_loss: 0.0610 - val_accuracy: 0.9892 -
30s/epoch - 216ms/step\n",
        "Epoch 8/10\n",
        "140/140 - 32s - loss: 9.7783e-04 - accuracy: 0.9996 - val_loss: 0.0608 - val_accuracy: 0.9848 -
32s/epoch - 229ms/step\n",
        "Epoch 9/10\n",
        "140/140 - 31s - loss: 8.1823e-04 - accuracy: 0.9998 - val_loss: 0.0574 - val_accuracy: 0.9848 -
31s/epoch - 219ms/step\n",
        "Epoch 10/10\n",
        "140/140 - 30s - loss: 4.3584e-04 - accuracy: 0.9998 - val_loss: 0.0651 - val_accuracy: 0.9848 -
30s/epoch - 215ms/step\n"
       ]
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "def plot_graphs(history, string):\n",
      "  plt.plot(history.history[string])\n",
      "  plt.plot(history.history['val_'+string])\n",
      "  plt.xlabel(\"Epochs\")\n",
      "  plt.ylabel(string)\n",
      "  plt.legend([string, 'val_'+string])\n",
    "  plt.show()\n",
    "  \n",
    "plot_graphs(history, \"accuracy\")\n",
    "plot_graphs(history, \"loss\")"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 541
    },
    "id": "ujWPxsde1sdP",
    "outputId": "a869edd7-2362-4abf-971c-d923ec59922c"
   },
   "execution_count": 22,
   "outputs": [
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ],
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYgAAAEGCAYAAAB/+QKOAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA
6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAgAElEQVR4nO3deXxU9bn48c+TyUYggZCENayKssgecWsFsba
41AWLuP6UXuV20etyvXXrVWu12tZ7rd5ab7FXLbe1VHGptVYFQbFXbQ0uSAZERJZkAsQAmQBZZ57fH+
ckDCHAADk5mZnn/XrNi7POPDPAec53Od+vqCrGGGNMW2l+B2CMMaZrsgRhjDGmXZYgjDHGtMsShD
HGmHZZgjDGGNOudL8D6CiFhYU6dOhQv8Mwxpi9VtzkUKpoAjsGRyc+9KEMHYRlRpGSo/KYnbnd7xWpNcgxeaXQ8fUD5+wEXy8bVJi
VTMYYY9plCcIYY0y7LEEYY4xplyUIY4wx7bIEYYwxpl2eJQgReUJEtorIyv3sFxF5REbIJxEoZZUUs+9KEfnM
fV3pVYzGGGP2z8sSxFPAjAPsPwY4R2z8sSxFPAjAPsPxMY4b7mAo8BiEhv4C7gBGAKcJeI5HsYpzHGmHZ49hyEqi4TkaEHOOQ8
YL46442/JyK9RKQ/MA1YpKrbAERkEU6ItsVWaItmqNEejzp8RZznSuqw0R/be17IciSpN0SiRSNv
zlUg0SlPLMRH3/VrPi5KWJmQE0gikCektr0Aa6WlCIGZfRkAIpKWRHpB996XFHtPm/QJt9qWlkZYmB/wt
VJWosife2O8VVSIR9/vGfKeW79gcje75TVr2tfkdWn7LPec5+7rKdAn9enbj0hMGd/j7+QRm3EBgU8x6uZu2v+37EBgU8x6ub
ttf9v3ISJzcUofDB7c8T+OMZ1BVdmxu4nN4Xq2tL4a2ByuZ2u43v2zgbqmSOuFrDkaJdo1rk2dQgQ3GTm
JKC1NWn+Hlou2X3F1BRMG9Uq6BHHEVHUeMA+gpKTk8Ga+P3mvr9MFqVn/HVo9TofMWFkM0oHc8RV7dbzB2X9VFw/AjJ+gpKQkhf67JJfmSJTKmnrWV+9
sZle3TLJ65ZBr5wMenXMoKe73LNbBr1yMsjMaFu+JdBMoEHVt6R5JhBJHZ1X9AGQfVxWcqxbTrgBdirw9CdlukQCdIBp6+3LIfsIVT6z+OvSwv6ZCxyx1Q7uy3DhZhNAoKNTYmZZCW8TbzCrLrEW0MvxFBFMFZF4GOVBiAvNUJkRvQfXBsyFH8aMuq9Sc9VWSVeSIr5Lko79/VU7LXKlNcdnYsd3hqr4cexZbjyhy3rlTcbz/Ka53dpRLUK8aMd48CL3sljNEQlq+wcDDgo+FB+KchFP8qSa2UWlRGS1KSdI9NS/9I1vtIJOdQ6w2DzD/ruKOj47lpwyqsIzKKjvbW+Er6VTlWt7SKZt6TGXJ+RGg9tz+UFs2VvSe6O3NpOeLz7pJBSsvSj8fn4eRyx3prKe
```

oPucWdM+kT142/fKyGNO/JzlZgZg78P3fncfe0bfsT09LIxCzvNf2Q7yTP5ySS+wd++GWXJy7emdfNKoE0t
JaY9s33rS94g60/hYH/r77/GYx56XHLgeEgMhBSzaJzs8EUQEMilkvdrdV4FQzxW5/s9OiMp6ob4qwadtu
1lfvZkP1LjZu280Gd7l8ex3NMbfDmelpDO6dQ152OptrwtTUNVNT13jAu8SMgNCzW6abPNxk0rq8Z3vP
vRJNJnnZ6aQHDr0prjkSpWpng3OnX1PP1tp6Ntc4F/vWRBCup7a+eZ9zczID9MvLpm9eNpMH59O3ZzZ
9c7Pp1zObvnlZ9M3Lpig3i6z0wCHH1RnS0oTMNCHTOkEmPT8TxEvAtSKyAKdBukZVK0XkNeAnMQ3TX
wdu8ytIE7+auiY2Vu9mw7ZdrRd/58/dbA7X73VsbnY6QwpyGDOgJ2eN7c+QghwG9+7O0MIc+uZm73N
npqrsboxQU9fEjt1N7KhrJNy63NS6vaaukZo6p7pm9eZawnVN1Dbse5HeK5as9D3JJCdjn1JLQ3N0n6qfL
3c20Lb6OT1N6JObRd+e2RxV1INTji6kT15WazJwXlnkZmd0yO9tjNc8SxAi8geckkChiJTj9EzKAFDV/wZeA
c4C1gK7gTnuvm0i8mPgffet7mlpsDb+UlWqahvY4N79b6ze5ZQItjnL23c37XV8YY8shhbkcPLRBQxxL/6D
e+cwpKA7+TkZh1QlJCJ0z0qne1Y6A3p1O6S4myJRwi1JpK6Jmt0tCaWRmrpmdtQ17tlW18Tqdkote1X3
DMhrveD365lFn1xnuaB7ZtJXOZjUIl2lFf5IlZSUqI3meuRUlfLtdfu0B2zc5izvboy0HpsmMKBXN4YUOBf9I
b1zWpcH986he1ZCN3G1llrSA9Jlq3uMOVIislxVS9rbl9j/g02H2Rqu57kPKli4fBOfV+1q3d7SHjCkdw4nH1
XoVAUVOOvF+TlkpidvPXRLqcWYVGX/+lNYY3OUN1Zt4dnl5by1popIVCkZks895w1lRJ9chhTk0C9v3/YA
Y0xqsASRgspCNTxbWs6fPqpg++4m+uZl8c+nDudbk4sZXtTD7/CMMV2EJYgUsX1XIy9+VMGzpeUEK8Nk
BtI4Y0xfZk0u5qsjighYKcEY04YliCTWHImy7LMqni0tZ/GqLTRFlOMG5vGjc8dw3oQB9MrJ9DtEY0wXZgki
CX1etZNnS8t5/oNyttY20Lt7JlecOJRZJcWM6p/nd3jGmARhCSJJ1NY38fKKSp4t3cQHG3cQSBNOO7aIb0
0exPSRfZK6t5ExxhuWIBJYNKq8t66aZ5eX89eVldQ3RTm6Tw9uP2sk508cSJ/cbL9DNMYkMEsQCWjTtt0
sXF7Ocx+UU769jtysdGZOKmbW5GImDOrVJQetM8YkHksQCaKuMcKrZZU8W1rOO59XIwKnHFXIv33j
WL4xph/ZGfakrzGmY1mC6MJUlQ827mDh8k28/HEltQ3NDOrdjZvOOIaZkwZSnJ/jd4jGmCRmCaIL2hKu
5/mYYS+6ZQQ4c2w/Zk0exAnDetuTzcaYTmEJogtZFNzCH/6xkTc/3UpUoWRIPj+9cDhnjxtADxsTyBjTyey
q00X8fV0118wvpW9eFt+ZepQNe2GM8Z0liC6idMN2AF6/cSo9u9mEMsYY/9nTU11EsDLMoN7dLDkYY
7oMTxOEiMwQkU9FZK2I3NrO/iEi8oaIrBCRN0WkOGbfT0Vkpfua7WWcXUEwFGZM/55+h2GMMa08S
xAiEgAeBc4ERgOXiMjoNoc9CMxX1XHAPcD97rlnA5OACTjzVd8sIkk7/YrGmATkZQ
liCrBWVdepaiOwADivzTGjgSXu8tKY/aOBZararKq7gBXADA9j9dXqyjCqMMbvBFG7GT79K3yyECJNBz/e
GJPUvGykHghsilkvxykNxPoYmAk8DFwA5IpIgbv9LhH5DyAHOA0Iehirr4KVYYDOLUHUbYfQh1DxwZ4/a
0N79r/9H3DOL2Bw278yY0yq8LsX083AL0XkKmAZUAFEVPV1ETkeeAeoAt4FIm1PFpG5wFyAwYMHd1b
MHa6sIkzv7pn0y/NocL3GXVC5AkIfuAnhA9i2bs/+3kfB0FNgwCQYOAl2VcFfb4Unvg6TroSv3Q05vb2Jra
trqodII2Rb9R/RiHNjkVMANt5XSvAyQVQAg2LWi91trVQ1hFOCQER6ABeq6g53333Afe6+p4E1bT9AVec
B8wBKSkq0479C5whWhhndP69jBtlrboQtK91k8KFTOqhaBRp19ucNhAETYeLlTkIYMAG65e/7PsNPg7c
egHd/Bav/At/4CYy7KHUuDM2N8P5vYNnP3ItiIfQeBvnDnD97D9+z3L0oeX6X5gbYvgG2fwHbvnBuJFq
Wd2xwkmXBCDjuQudVdIzfERsPiao311URSce5qJ+OkxjeBy5V1bKYYwqBbaoaFZH7cEoPd7oN3
ExgFPAxNUtXl/n1dSUqKlpaWefBcvNUWijLnzNeacMpTbzhp1aCdHI/Dlmj2lgooPnOQQaXT25xTsKRUM
mOQkhty+h/YZmz+Bl2+E8vdh2Klw9kNQePShvUciUYWyF+CNH8H29TB8Ggyb6ixv/wK2rYeaTUDM/5v
MHm6yGLp34ug93EnIaV1sIMX68J6L/nY3CWz7wvmONeUc8LvlFMBni2D935zj+o6FsRfCmJmQP8Sf72
OOiIgsV9WS9vZ5VoJQ1WNGqtRhHfmsZVjp+Rmc6kEtAiRN++YNNXgPTwbWCv/5AbYGjA7l6Pdvuk3f7LSkALSuYvLd4n4oTchhZH4dkz8n+Fwk6OBSEmbFVc
Mcbv1S+rEDTlfGp4UkUd+hRH55Ln2PQ5nsSUJn+XMm+fuPHv/qIdYfFL0FMgFENzUy87wlk7dCm1sfJuu
J9K9Gb0u09DMcxdUf0KGqP+yZ6CDFuqdudN1WnGcOtsB+e/Y4C+4I2H1rR4hLPHU/J+CX2X2nV+9vCGGjfw
x6fUNMjkHhoOT0DOFIw/gNPVKf9ZhG2Fv7cOq8dq7V7sRq2cLHcSS8WpTMIzRjA9Xu0mt0TC2OJHtpiF9uiy
n7NffU9W2yBd7HqLmBWrYP6bF1g7sFTSf2pE96H2q2rZwDzyYeJi2/NmrxWkjQ5SpFGBNOFN5wfHlmE
x8nfGp1YHWY7lmfRVWHxKd0nMVaFN+vY5VzKtCeG6YVU5GiK/o8cXF4mF2pO8EIoFmQa6xc5vCdE+Kj
5RELJk+2UhqsZMH43D9FNSQaULXv6LpBZOi6HkwzNH7cAzkIXmUWWp1X88Ssr6ekhA1X5J7lVMZLnsq
ETSyJiEL9aMzPHrJ/PwE88cVaHfWX7mfsMXmGywXkJMVpnYDvtE19Vd6bpW1nBhQvXQOSbDbwB6eDZX
VnxQjN0mdl/ZjY9H2A2CCPBGCMjfFTpOQ3+7iHPeW+PmzIEyzchlf9WXFXXWA/Ljjld6Z1DWO2dA3T1x
KWxbLtgNXtVzs6jrwNsCeAZoaA2cRNeL0wOChSiK3pGvDBWRxoXzXuQC6aGsebYlDYWiW3B+fPvo0D8S
Pk7mWfLFHT+NXD2rMWsqy/ZmP4T8JjnWkydETQXL8a+f6i2gdgkrw6xgAqwu2mEz4LuUn91lKUlBiaBs
ZR2tfM2MUNN7dzCk8Yw6/7I2XyTTMn9EeT5cqD99sUzwNGe3F0+V6WoZxCRoR9RQzONfuaS+o/KRS8EX
8fx/fo71PZo/fRwVh6J6O4WUmSKTwHr8Hc+yD4Lmb4H4Nhac27kL+d/rX0ZtPA5nhPz0n/Aa4AQvwg5
ES4d3SwpeRZ3Pt2u0wsKkB1lf3vIE9LZFz1LcFQaXYxD9AGpLSnm3W6bwdWBaVf2FnSOXwJ6o6v9Gl6T
hcsi6xWXZ3zCFx5ZrsiKCRA51xK9MKDQ3S4A4QA7cR62QYBt6pLHJHMAcYEK6lLPCA8GvZDqCKx3Ltx

x51nPxJFIIH/q4rAoOOd1zd+Ahv+zylZBP/ktGdl94RR5zoli2Gndq2SUIpL4H91ia/1Cer+CV4vKwITLoFjvuE
09v7fw7DyBTjr53CszyOkbCmDRXfC2sVOr64L/8fpkpkIVWHJKC3gJIFhp8JZD8LnS51kUfYifPi/0L0PjDnfS
RbFU+zvyWeWIHxUFgqTJjCyX4IniBY5veHc/4LxlzrPTvxhNoz6Jsz4KfQc2LmxhEOw9D746GnIyoWv3wtT
5nZeY605uECG0yZxzNeddqHPFjnJ4oP58I950HMQjLnASRb9xydm9VqCswTho2CohuFFPeiWmWRF6iE
nwT8vg3d/CW/9DD6f4nQdnTLX+6qShlqnBPPOL0EjcOL34Kv/al0tu7qMbjD6XOfVUOsMGrnyOXjvMXj
nEadbdcvT231G+h1tyrBGah+dfP8bHD+sNw9fPNHvULyzfT385WZYu8i5CzznIRg4ueM/J9LsPMz35gNO
d9HjLoTT73T6/pvEtXsbrPqzkyzWv+0MGdP3ODj2zK7VxuW33P4w/uLDOtUaqbug7bsaCdXUJ377w8Hk
D4XLnnUaJP96Czx+Oky5Bqb/0GmcPFKqzt3m4rucYUcGnwyX/BGKPUhCpvPl9Ha6IU++Emq3OP+OVj4
Hy37ud2Rdy8CSw04QB2IJwictDdRjBqTALHIiTsPjUdOdLqb/mAfBl+DMB5xhGQ63brliObz+706vmIIRcP
HTTtdVq6tOTrl94YS5zqu5Yc8AlAbw5t+8JQiflIVqgE6eA8Jv2Xlw1s+cO52Xb4Bnr4Kjz3B6O/UeFv/7bN8
Ab9zj9KvvXgRn/4fzkF7A5vNOGdbZoFNYgvBJMBSmf89sendPwQeEBk6Cq5c4Q5wsuRd+dSJM/QGcdN
2BH5iq2w7LHnRKIBKAU//NGTguK3f/5xhjDpslCJ+UhcL+TzHqp0A6nPhdZzC3v97ilAhWPOM0Yg85ee9j
mxvgH4879c71NTDhMph+B+QN8Cd2Y1KEPYXig/qmCJ9X7Uz+Bup45A2A2f8Llz7jDJz45Jnwp+87vVdU
nQbJXx7vPKU9cDJ8529w/qOWHIzpBFaC8MHqzbVEFUanQgN1vI75hjNE9Fs/hXcfdXom9RwElR85k9Jc
8YLTyG2M6TSWIHwQDLX0YLISxF4yu8MZ9zhDiL/yA2dAvfMfc9ZtfB5jOp0lCB+UhWrIzU6nOL+b36F0T
X3HwJy/+B2FMSnP0zYIEZkhIp+KyFoRubWd/UNE5A0RWSEib4pIccy+n4lImYisEpFHRJKnc3uwMszo/nk
k0VcyxiQhzxKEiASAR4EzgdHAJSIyus1hDwLzVXUccA9wv3vuycApwDgOOB4IAnmtoRIVFldWZsaD8gZY
xKalyWIKcBaVV2nqo3AAuC8NseMBpa4y0tj9iuQuaApwDjgOOB4IAnmtoRIVFldWZsaD8gZY
gsClmvdzdFutjYKa7fAGQKyIFqvouTsKodF+vqeoqD2PtNC1PUFsDtTGmq/P7OYibgaki8iFOFVIFEBGRo4F
RQDFOUpkuIl9te7KIzBWRUhEpraqqaru7SwpWhskMpHF0nx5+h2KMMQfkZYKoAAbFrBe721qpakhVZ
6rqROAOd9sOnNLEe6q6U1V3An8FTmr7Aao6o6T1VLVLWkqKjIq+/RoYKhMMf060FGwO/cbIwxB+blVep
9YISIDBORTOBi4KXYA0SkUERaYrgNeMJd3ohTskgXkQyc0kXCVzGpKsFQ2J6gNsYkBM8ShKo2A9cCr+Fc
3J9R1TIRuUdEznUPmwZ8KiJrgL7Afe72hcDnwCc47RQfq+qfvYq1s2wJN1C9q9F6MBljEoKnD8qp6ivAK2
223RmzvBAnGbQ9LwL8s5ex+SFYmYJDfBtjEpZVhHeisgpniI1RVsVkjEkAliA6UbAyzNCCHHpk2Qgnxpiuz
xJEJ3LmgLD2B2NMYrAE0UnC9U1s3Lbb2h+MMQnDEkQnWeUO8W0JwhiTKCxBdJKyljkgrIHaGJMgLEF
0kmBlmMIeWfTJy/Y7FGOMiYsliE5SFgpb9ZlxJqFYgugEjc1R1m6ttRFcjTEJxRJEJ1izpZamiNoYTMaYhGIJo
hMEK90GaitBGGMSiCWIThAMhcnJDDC0oLvfoRhjTNwsQXSCYCjMqP55pKWJ36EYY0zcLEF4LBpVgpVh
q14yxiQcSxAe27R9Nzsbmq2B2hiTcCxBeKz1CWobpM8Yk2AsQXgsGAoTSBNG9O3hdyjGGHNIPE0QIjJD
RD4VkbUicms7+4eIyBsiskJE3hSRYnf7aSLyUcyrXkTO9zJWr5SFahjRpwfZGQG/QzHGmEPiWYIQkDwK
HAmMBq4RERGtznsQWC+qo4D7gHuB1DVpao6QVUnAOB3cDrXsXqpWBl2raquWBBgraquU9
VGYAFwXptjRgNL3OWl7ewH+BbwV1Xd7VmkHvlyZwNbNbwg02BpMxJiF5mSAGApti1svdbbE+Bma6yxcA
uSJS0OaYi4E/eBKhx4I2B4QxJoH53Uh9MzBVRD4EpgIVQKRlp4j0B8YCr7V3sojMFZFSESEmtqqrqjHgPyZ4
5IKwHkzEm8cSVIETkeRE5W0QOJaFUAINi1ovdba1UNaSqM1V1InCHu21HzCEXAS+oalN7H6Cq81S1RF
VLioqKDiG0zhGsDDOwVzd65mT4HYoxxhyyeC/4vwIuBT4TkQdE5Ng4znkfGCEiw0QkE6eq6KXYA0SkM
Cbp3AY80eY9LiFBq5fA6cFkT1AbYxJVXAlCVRer6mXAJGA9sFhE3hGROSLS7u3xqtoMXItTPbQSeFZVy0TkHVMh
G5R0TOdQ+bBnwqImuAvsB9LeeLyFCcEshbh/G9fLe7sZkvvtxl7Q/GmISVHu+BbuPx5cAVwIfA74GvAFfi
XOj3oaqvAK+02XZnzPJCYOF+zl3Pvo3aCWNVZS2q9gS1MSZxxZUgROQF4Fjgf4fvqmlu+uPlLqVXCJrG
UOCCtBGGMSVbwliEdUdWl7O1S1pAPjSRrBUA29cjIY0DPb71CMMeawxNtIPVpEerWsiEi+iHzPo5iSQjD
kPEEtYnNAGGMSU7wJ4prY7qequh24xrOKgpvDQBkZD3HiVIZ7GtVFRTpUCNDvTPNzpwvBnibYBlR86aP6
RHrf3BGJPQ4m2DeBWnQfrX7vo/u9tMO8pCNYD1YDLGJLZ4E8QtOEnhu+76IuA3nkSUBMoqwmSlpzG
8sLvfoRhjzGGLK0GoahR4zH2ZgwhWhhnZL5f0gN9DXRljzOGLdyymESKyUESCIrKu5eV1cIlIIVSkLha39wR
iT8OK9xX0Sp/TQDJwGzAd+51VQiSxUU09NXROjrf3BGJPg4k0Q3VT1DUBUdYOq7c2V1YiausoqWB2k
oQxpjEFm8jdYM76upnInItzrDdPbwLK3EFK8OIwMh+uX6HYowxRyTeEsT1QA7WEF
gozvLA7OZlxj4NojDFd0kGvYu5DcbNV9WZgJzDH86gSWDAUZtKQfL/DMMaYI3bQEoSqRnCG9/
3IxU76qz9wRiTFOKtB/lQRF4CngV2tSyq6vM3Nq7Qzv8ShDEm8cWbILKBbcDpMdsUSPkEMap/nq4hqfUJjjg3ugiqG8hHIp5LW4Ex5NpLAtr8LXrv8txDkV3hfDTQq1/Y7KcXxNNW0dD2dD0kGvYu5DcbNV9WZgJzDH86gSWDAUZtKQfL/DMMaYI3bQEoSqRnCG9TYHsWN
3IxU76qz9wRiTFOKtB/lQRF4CngV2tWxU1ec9iSpBtQ7x3d8ShDEm8cWbILKBamB6zDYFLEHECIZsDghjT
PKI90nqw2p3EJEZwMNAAPiNqj7QZv8QnHmoi4BtwOWqWu7uG4wznMcgnGR0ljvLXJcVDIXpm5dFYY
8sv0MxxpgjFu+Mck/iXKT3oqrfPsA5AeBR4AygHHfRF5S1WDMYQ8C81X1tyIyHbgfZ0pTcB7Gu09VF4lI
DyAaT6x+KguFbYA+Y0zSiLeK6eWY5WzgAiB0kHMmA2tVdR2AiCwAzgOCMcaMPOmgGLgQK3IFGBnIlLmTNo6X5TAyizf7brVT4wnB2nL8K3nXg+ELvHs7smd+D6SaT2dBjTJ6QMbLXq2f+JGcl1F9YdUsVQQpMpTQ7Evw+
DyAaT6x+KguFbYA+Y0zSiLeK6eWY5WzgAiB0kHOmAGtVdR2AiCwAzgNiE8Ro4CZ3eSnwonvsaCBdVRc

BqOrOOOP0TX1ThLVVOzljdF+/QzHGmA4RbxXTc7HrIvIH4G8HOW0gsClmvRw4oc0xHwMzcaqhLgByR
aQAOAbYISLPA8OAxcCtbo+qLmnNlloiUbUeTMaYpHG4w42OAPp0wOffDEwVkQ+BqThPaEdwEtdX3f3
HA8OBq9qeLCJzRaRUREqrqqo6IJzDZw3UxphkE+9orrUiEm55AX/GmSPiQCpwGphbFLvbWqlqSFVnqup
E4A532w6c0sZHqrpOVZtxqp4mtf0AVZ2nqiWqWlJUVBTPV/FMWShMblY6g/JzfI3DGGM6SrxVTIczsND
7wAgRGYaTGC4GLo09QEQKgW3ufBO34fRoajm3l4gUqWoVTvfa0sOIodMEK8OM6p9HWpoc/GBjjEkA
8ZYgLhCRnjHrvUTk/AOd4975Xwu8BqwCnlHVMhG5R0TOdQ+bBnwqImuAvsB97rkRnOqlN0TkE0CAxw
/pm3WiSFRZVWlzQBhjkku8vZjuUtUXWlZUdYeI3IXb62h/VPUV4V4ILNzPuYuAcXHG56sN1
bvY3RixBGGMSSrxNlK3d5wNV+oqcxuorQeTMSaZxJsgSkXkP0XkKPf1n8ByLwNLJMHKMBkBYUQfmwP
CGJM84k0Q1wGNwB+BBUA98H2vgko0ZaEwI/rkkpl+uL2GjTGm64m3F9Mu4FaPY0lYwVCYacf6283W
GGM6Wry9mBaJSK+Y9XwRec1bLH1nA9X+5ssPYHY0zSibdOpNB9gA0AVd1OxzxJnfBaGqhtDghjTLKJ
N0FE3eG3ARCRobQzumsqapkkaJSVIIwxSSberqp3AH8TkbdwHlr7KjDXs6gSFmohsG9c8jLzvA7FGOM6
VDxNlK/KiIlOEnhQ5wH5Oq8DCxRBENha38wxiSleCcMuhq4HmfAvY+AE4F32XsK0pRTW9/E+urdXDip2
O9QjDGmw8XbBnE9zrDbG1T1NGAisOPApyS/1ZtrARgz0EoQxpjkE2+CqFfVegRyVLV1cCx3oWVGMoq
agBsmlFjTFKKt5G63H0O4kVgkYhsBzZ4F1ZiCFaFaGKeieSZ/cLL9DMcaYDhdvI/UF7uLdIrIU6Am86llUCaIs5
AzxLWJzQBhjks8hDx6kqm+p6kuq2uhFQImisTnKZ1t22hDfxpikZaPLHaa1W3fSGIla+4MxJmlZgjhMLU9
Q2xAbxphk5WmCEJEZIvKpKVkXKiKwVkX1GgxWRISLyhoisEJE3RaQ4Zl9ERD5Y5yXy95GefhKAvV0C0jwLDC7n6H
YowxnvBsVjgRCQCPAmcA5cD7IvKSqgZjDnsQmK+qvxWR6cD9wBXuvjpVneBVfEcqGAozsn8ugTRroDb
GJCcvSxBTgLWqus5t0F4AnNfmmJHEnd5aTv7uyRVJVhpQ2wYY5KblwliILApZr3c3RbrY2Cmu3wBkCsi
Be56toiUish7InK+h3EesvLtddTWNzO6vzVQG2OSl9+N1DcDU0XkQ2AqUAFE3H1DVLUEuBT4hYgc1fZkE
ZnrJpHSqqqqTgu6LNTyBLWVIIwxycvLBFEBDIpZL3a3tVLVkKrOVNWJOEOK0zIxkapWuH+uA97EGf+JNu
fPU9USVS0pKuq8KT+DoTCBNOHYfrmd9pnGGNPZvEwQ7wMjRGSYiGQCFwN79UYSkT4i0hLDbcDj7vZ8Ed3
u+iGS1HAOcAsQ2bvuqLBTmqKLuZGcE/A7FGGM841mCUNVm4FrgNWAV8IyqlonIPSJyrnvY6cCnIrIG6Avc5
9AXuc7ePAkpF5GOcxusH2vR+8lWwMmzPPxhjkp5n3VwBVPUV4VBVPUV4JU22+6MWV4ILGznvHeAsV7Gdri27
WqksqbenqA2xiQ9vxupE04w5D5BbQ3UxpgkZwniELX0YLIqJmNMsrMEcYiClWEG9Mwmv3um36EYY4
ynLEEcImcOCGt/MMYkP0sQh6CuMcK6KpsDwhiTGixBHILVm8NE1Z6gNsakBksQh6AsZHNAGGNShyWI
QxCsDJOXnU5xfje/QzHGGM9ZgjgETgN1HiI2B4QxJvlZgohTcyTK6sqwPUFtjEkZliDi9MWXu2hojloDtTEm
ZViCiFOw0obYMMakFksQcSoLhclMT+h5+h2KMMZ3CEkScgqEwx/bNJSNgP5kxJjXY1S4OqkpZqMb
aH4wxKcUSRBw2h+vZvrvJ2h+MMSnFEkQcyiqcBmorQRhjUomnCUJEZojIpyKyKyVkRubWf/EBF5Q0RWiM
ibIlLcZuiNxy0AABLLSURBVH+eiJSLyC+9jPNggpVhRGBkP0sQxpjU4VmCEJEA8ChwBwJjAauERERrc57EFgvq
qOA+4B7m+z/8fAMq9ijFdZqIZhBd3pnuXpDK3GGNOleFmCmAKsVdV1qtoILADOa3PMPaGMaGCJu7w0dr+IT
Ab6Aq97GGNcgpVhRln1kjEmxXiZIAYCm2LWy91tsT4GZrLFwC5lIlgImnAfwA3exhfXGrqmti0rc7aH4wx
KcfvRuqbgaki8iEwFagAIsD3gFdUtfxJHSqqoqTwJcVWlDfBtjUpOXleoVwKCY9WJ3WytVDeG
WIESkB3Chqu4QkZOAr4rI94AeQKaI7FTVW9ucPw+YB1BSUqJeflmWOSBskD5jTKrxMkG8D4wQkWE4i
eFi4NLYA0SkENimlHgNuAJAFW9LOaYq4CStsmhswRDYYpysyjKzfLj440xxjeeVTGpajNwLfAasAp4NwLfAasAp4RlXLR
OQeETnXPWwa8KmIrMFpkL7Pq3gOlz1BbYxJVZ7221TV4BX2my7M2Z5IbDwIO/xFPCUB+EdVENzhLV
bdzJ9ZB8/Pt4YY3zldyN1l/bZlp00R9XaH4wxKckSxAEEQzYHhDEmdVmCOICyUA3dMwMM6Z3jdyjGGN
PpLEEcQFkozKj+eaSlid+hGGNMp7MEsR/RqLKqMmw9mIwxKcsSxH5s2LabXY0Ra38wxqQsSxD7EbQnq
I0xKc4SxH6UhWpITxNG9O3hdyjGGOMLm+BgP4KVYY7u04Os9IDfoRiTkJqamigvL6vUAyQnZ1Nc
XExGRkZcZ9jCWI/ykJhvjqi0O8wjElY5eXl5ObmMnToUESsJ6CfVJXq6mrKy8sZNmxY3OdZFVM7ttbWU1
XbYO0PxhyB+vp6CgoKLDl0ASJCQUHBIZfmLEG0Y08DtdtfVgMuZIWHLoOg7n78ISRDuC7iRBo2ySIGNMCr
ME0Y6yUJhBvbvRs1v8jTnGGJNsLEG0Y1UobFOMGmPi1tzc7HcInrBeTG3samjmi+pdnD9xoN+hGJM0fv
Tnsta2vY4yekAed31zzEGPO//889m0aRP19fVcf/31zJ07l1ddffZXbb7+dSCRCYWEhixcvZufOnVx33XWUlpYiItx1111ccMEF9OjRg507dwKwcOFCXn75ZZ5//nnmz58nOXXwJTAdsgQRD3eEtcmtmZ9CbGQXrVNrtYdGXfI2eQPrVR+QY2+FYRCzsy6/8c0d/1qrKO6xdTW+WYuaaJeO+Rv/rrhhRcxSz01R9O9r7/+i7zc/0+3KnpuHPb8mjHX+vmOJ8C0kFl6QSuxdMVRcb5fLXT/e9qoDp+NsOvrdYfXqGd2ZIFzvRvW7uUXTzyUFnpn3IVz5fZx9/OJ7HtlnwPC+F54cbxDGPOWF1EH4ev9vnzxr5/Dui/PcYePjuQJBtvXivUx7WWxsWXz/8rgpqk9H6o7X0+cgszNYu+aSwYRi+4zS8jRpq3/7zjn1+vbpmUvGFHVw4iJVg6ccftiVgv+Luzfvu0NxLc/N7zPJ5V/ebQS4EXXXvjWlZ5ufLF64KdhuKYrMvm/P8C5R/h7h2/lWY3jHhzhDP31eYbD/n9TMi+WiN6xg+bUGlXj9Bvw2/Ly2tz7jHdfS/c9Y1HXh6xHVyKPg8e+zbfnyspsFRfTZ57J1LueM7Y6ZZLP9S+cvk6RrynfrN++7qj/1PH/ffl/5F/0OPvxeK7qGvn3v5TrvuJCfP33/b+/x33h/4//8A/vff/3+d+//9/A/0/PyrR4bVMLEehcaVvqNV5e1UKEV+h0hgKumG1I9u0v/8+vJbzQeJ+5BwuL65/0/LJyY4a+4K6WCP/4/qf4DyvLIqRVW9sbbmuYLQ/SFfQNlMjnAuLbWeqhaOOvQBeVvjZG6pGv1O8bw1SGcJmWhWVVQFCvyLiiyezw/yfOPG3lADAWgqHvo9RBSiP1vQKMbhI/1fhbeazjGYK+Y2vk+S+XOgYPb4mjWM1PkyfTc++/uD/+Y83OvaJcuzYFd/F2bWrdW5bvn/mdx3ucSrWPSBM5Pf9mNKPFA6ax5D+CZtPYt9BcC9k+ZvZjdZ9+TlQXQtq0UJyiWUc7ylcM6oHX8SKWV0Ue+N//fRRrCzL5CEVMmjgRfpWPdd8PfWK/Tdv9mwG1szF+Bo4fmnqeNxCZwL+gd9Bp8zJQCoFM0HaOO/EZlW6tGpGb9eZx8Ac1/K93+fHuS9Cv7AGbFK3wOqJ0+6SlFHZSCNBrv7zrOKEGXxFLl33L8qn2c0/ZUsu5N7g/b9w1z2Ov4NHcY4UnbjMHbMyeshE+Qj4PKrmnoY3Hib53KsXJpHtuz9B2IhaHE9vgXyMHpyOzB9qd/d/LlKOYI06t28dZXSB70rd9uDmO5SBsw7bYIhfGEmCB2dZ6/xn5Bu+QL1q+uDEmOzQwJqfmtwyHT9Iko/8NaG9wvt8ZZEE1xF1tqCdxR/6t0u6q8/4mzu+V6Z5d2Gy8yaz3ZV4mE/mw2qUYaXTumZqekbQ8dQA9AsVv3Lm/s22Wp/qlsEcStxsi/KZ8jb2N4/4QJPsmG91LdOZ9dAT8QFczL/ZbKb/ioaM+TSsaJDf7qtmEDeLEGoZA1QKlVYHkn9y2Tbvc2iWbMC/ijCyUs+aTMWiDQkm4c7IPeRDU8uXEjU8JbH5mY1OPNrQcv9+HN+rf6Fu+0d+y8cgfmSofdL1IIE+VLlb58F7Hp3edJIuDe6GMKc/oTMrESnvnOswHz5E+7wEe8CcrU6zAVg3tlCY1lA6w7LTzNmsbAGVGfHkLVgCg5G+/jX4lSVtEB6yIxQ3AwqdORCpOmw+/S7XTiZqDKnCVdD1xaFODuhgT35AqAUtLPTSNt4iKmuIiIlpP2+0ZQ7LDHWi+WkNXTONRF6xhc3HBv8iN/eNJqK5dW3g0w/MEt5+gjUx9DdXT03Xbf1VgKGGQ9uGspz/QfuTzXPKRNS/e/SJ6O9WBaGUDr3GpLb1nY4V8w26NqW35OXh+lUWSHv/ajSqObbjJZrkr1E+X4pJtjCE+ocmLOrfnkXUV7bhmGn5wxzY3/cbwNRd+Wmfq5QS/pflcEM+o8ff6/xsfRyJbgV7+LaoY/XOd3PJppYDcU6wZLxyNmp5xHo23o2dqnHzSaxjf9Hn3gHuDrLAcb0eEz5n5kdoelNyL2Qw8m9/Rqd8O/AX4PUcKlRU6d8+8SUpnSs4SI0PfPdw2zJxQEDEHfZZPcnvBTKd9Ks6Q6rYzWLg5nxfDRmeX9LvxWkP/GmRc6mESvpvtvG+ZsIRqwp/UF5PdW9nYfJXSfDl2fYGK7w5ek5C5H2Th+ZdMSfZVGZbfm2cnLdU0VDx9xt+MQDG6EM7xEzxLtvNPEHzfv+v/b5fDLOLMsuEQ1xy1jsAqEwFSn2qaDE9Db/jO6/ni0qVr0mpGhdVj4OD7sw59PlC9W0ZrCRF1z3qZtvVcQ1gSs7HHubF8jS/JaRXLtqL4XWbMRXW4ujuPpuoZsEMWm3VmuqdHwqqzBCsfAhUmuZ0D8qQEgVXfeM6a54XGh5hX45ZgFaD/gMFAvrHNk6fKvDcdUzXJZpmlhRbCQ3NzD8O5dpb2+uH6PSOdw2HIyHr3sLTE8hOP2mvRs6xsr/Hr8P3V0JXfg+kf3d7+1ZqZc5CEvkxT+CA6ANsSKumUu9GaaFmFJJE0QQN5RwE04ZPr4MIZ8b91ITHqRkC85C16/W5/09ZmIpsZx3KvhT1Ic/5aBWzMvsG5HnVlNvVP7Wu2ctiQsmgB6OJrO6M4A6WnPmMJn5eyjaX0NPR4qLD5D4TBnRiE4ipeq6rqMsF60HvmIQaIs+hZSrpPyuy1EbMLxXeZ5wQ4nXnW/HQmPwkhWUQ8qzVdR1s4TMd4x0uk2JKoAtOSv3sTMfaXTk9MSdiTqCl+WDUh1jCg3CR8l5DUOKNxHemBFx4yvF/tYFbY/QftS0Fr2LcbI1WVOxz+xSVV5WWbL1ZUb+/l5vSHV+H1WzlfVUZ1AKjWXGRgNbDR8A8/k6O5qJhbKd14T45CPuTa4t/s3lXt20XiBRIcBh0+LS6LTp8+D8OmMlM66Mh4+NJKsuxsl3vJpAe7Wst7HrBQHkJr8cefetyAhcqhkb52Zd+L/qHUqoZCyWVNv2y+zx8a6qGAAdscbCeF7HXsxOA1j1+1IM5nktkV3xB9n9tBHpZ8wu+JuHK8uW0+aK0uyhyxXGbfK1fBrH5dGv67jDmZ/XYTx7oeoaZYrMoP7d2sIEfXEPnwVu5ezjq40nkNPWsUUpq/yjt3fGrxRDF0qmU0OY5uS8bTd9rnXGWQPeaJoZu+0pSXtzS/sR3CD+ahOaL5qp+xCvXBoFFvgObsv0LW4Z7cPKX+NUYhXhz/eQkrKJAdN6Q2iUtsDAZGOErWpDTbnxc6+t5v60hWd8y7JNdGGqaM/+iu4zO5tEBq6lvvMEJXI8XvN94D2rDaZKqhA4nmL3rNzxv0d8CUxjppcvNmeq7SOCUXsUIR4fCeRzbCvQNqLHVqxXbBiHCPoaI/ymvDVh92S4LMWm6R/4j1GBjSdXiimk1uyQb8Ae8d5qJv93jtl3nQw2aZfP73qEdd2bd4fGpLSIbsQcGQhbRHq9TbkUwDkAnGA/j+wxNz6mZJtCWqUGiaKR2NlBF8Bu6W4hcMZ5NNyYc0dX5Rb8s59RFs34IvxFiY+oHbU2oc9Yfjgc5P/+Xw5D5H0xmD0pNjlLvNvLvU5fJmHmzn+e2SdOxVXuYHgAi+VX1+uSI+pPWvA0Ux98mVDlCkMgxoKh1tV6rf9Fd3pA2rtvvYAdu3qIYFpBmG6TzbVDpRLo1Q8d+4yeFPzmEDWUxbAcYIdZUnDCdaIWsI3TFC/a8u3YrzG1B0Hb1o97Kx0z+DjRbIpTdKDI3Kxi1BySlGWSzlXe0M+i7T2/4kfz6TPLQxiz8+FuxfSqvA0ZZkRqXvmKc8Y8I2ej+R0Ll1m1D6aKvNx/FtTGBMP+9rLqrkHfZ4jm2sR9nXOd0mvxxgN7cpNozsxDJC3gIFczhL1XIO/+pBGnF+X3Cu6NJcgSmF0BDFVC7Dpw/3AA5BpOw8+iOGcvdsgFj+tGWTrhJYRgpfjsDDAjCIBzQEbKSxUXgn9I2x+QRjbn3WSs+mi3DTqnvdSKtQ3okj4nG8wbK8j9edTnGxC5PAtFXUbIm0X8Nl/WVUZf5l+Rj0h+L1rDRzUmRlOy5NjMZq+2dQHO0d1Jbi6dmYaSq7ZqFWmZNv0aRT8eV0YqRJf7fYUJVF5BdkRqAzb4tX2Sex7+eO+IeVqRufSUI8ib0SuLzLZ3mJrETu+NDtZ9mnM1BqDWKJCcsuohb0w9QknA45OUbR24dIA7H+6tdlJh+qA+nORo3atzEm+k5LJqT3JXbbIGRQwK+sI8+h9hs21PeGcGSE9rMBkT8UMgmpU3jKUvhDg8jqKf5Zb+rDKfTJPsXTB9HuB7yrfZOKIgntrGIcJ8oyRA1GcmPK6eGOlqfGqu6PbG6NtCxPwKeR0ncqwa+yAoWU3hm9/6DAHahjcavPyBjKy5rhU7zAnJHsQGhB3LVJq2u4g7OiqgUcQ3BVbXHSXTtcdJjMGs/+cz5VZ0QwH36Z59v2i+rDfGT+oAxBuM6+Kqdgh1ZyNV3o3Gs29QQGvBRFSjCJbP0J0SZnC07mAE1sDa4Y6x5o+JIz+3pHTpLshvTOR2C6bpFGVJG2pGVNvC5xd/K3EzR7ZkfaMKoJ7oEdwNnmnlCWdYWoIKRhjWzfKWJrGqL90w95+aTE9ntSRwvTd0/8+pUcLakoS+K3PA8dSi3RwVx+u4VYRCWWMUsXrNR91Cs78NT9afTDehQxqdWh+U5n1lfKeCCnhKTNdqPjC9vJydWmN57IJb0MY2kJKwMCb0B7AJcyyzfzERPk3XNkg/HDgQUP2Nmb8EqmRw3MTwJU1PzS5KCT8dM59tkXCPb+e6Cf5Ju5fKndL4CGSTYv52Q4ZtByhOgfrW3u4x/CvM+Eb0hjjfUZfj1PDeu8+Wxu1n/NPwsVc7cTeXEq8Ji6ylKI+hspqxBvnNiQ/GsXfOsb7mfM3iKcwMDOFm2bf1H+lsqkY+pg8UZsNC1eSRmu/VhtP8bIiPnhsOUHAV+Pv0xQ8csIgxLDGz3WVE1k7JhRuqeoqiv2x1pn1/6bIcFw4eKTHNPTDvF/U4LqKqLdzP6dOjd+1SbvjCjRWwgGpSH9H2CPaj8nTQNMI8t5QC5WJPWtMY9WPxjpvGLvz6ycmJsfKCT4Y/TRPHqIB45Tpo+fWpMDvYlY84qJAcVDjoP8sd4sfNzWpfAhwv1V2uvH6X7p7TN3ipe8ek6oDz4ob33W7v4Uie5nt/X88UpQ7Znb4TrE9G0aVbSf33CGUsSpAbc9NnWZM/h9i92SdxYu+kEX9kPmrBd2wBZPP6sGTDO/b5vkB8rBoSAuO0Zo/MVL+HvfKLO3zXGG8qqi8v+MaUmFWoJzShGL5nXw1UYS9EZvRDp79HZ1oOsYqb3tCkcrZyCTt0rrH6bDc6Q+qx/iDaoPCKxPs30dqOBp3ky0bNhuZe/PkgzHAAdAr6mcqpKeaM3UBvD/ln3MK9M4RcL8S3SWDCBLy+1rl6ej5DhsA+ahNDzK+kBxvU7bLtN08kGrmvl2YmUH4TOHK6XORZPB5XSqEJKuzJDWbl2mM7hcmG7nqQIaNCB6dAp5A5aY7HT4UQdTDiSbo8rD8TF4X+0bDNm5qTBbhK7L+2PD0vfV+ekoPYkT1GqFDmD2qAwTxtwszU5BqsGgUvMTuGtFnFoa9Gk36z3RE5oCoLE8FGKNb1aWrWrJc7w/0ACE4+L4xaoHXqHIlsz5PJyZNGOPu3pY18DJdJUxu9ZsLTBbsfZSuE6vdOd49qx9VqtFAIIpiE5ZRkPIPCljVgMt7C+sJn6LkZ8C/8kKQxJVHhVfTqxHsXrbJVEiDe4xTj4YUBgeNW+cJTsM9HrHa6zhMlF9sKSJk4TQ6zp4ox3mcA8vgL9zOUL6j9r4AQxaVf1rzEZc8TmTTrQ3BkwjMFAj8NGxQMbPsD0p5lqzMZSjXoBSvv+T6HYaO4FoFZ9UKCyP7WG4upH9FSmYn2SsdVu3HDrOXqeoCj0GpYIcT0nJHqaHV9spfxfUbx4Bpt6GvbBMxWNPHxJyLQ8GtNPW7P9wSc3D9RrlNrGImVnxhW7QgkjcfPBtuQu3EyLNhu7lUe4LkaW3bP7fk8V87/4WxSF0yKMd5YfVVEdDn2GlKMsTTFKt0cPfH8knMu2BZxH0wwEMi5lbD43kBvlHuFqwOqC7Mg/LR8nKXn1xKcddsSt7lrzCRkvu6k/T4OjKzsGzmUDF3P3l8vtGhqeR6d7iYJmENmtFiAH5WcvgYPYaoDSzFg0XSqt1WpT4z+8kBBWJEHXIevjIchXyhfYBoXIV9zOIJCmT4pvdqWN8pBHsmtpYWc3+7b+NcLXvBbNHaHJKjlwAyKv7sgaJ2eN6Adu/r6ZW6PBKgm9n3v5p8U/u8QfU+0w3K1zIsSjo9IJUEuGCyr9m8FtLGcHgcHqr8VUhtUQ2PSKF7nY9qhjcP1zJhY2nRaEDRTqvTjxL79vAbRFu67ISv1N6TXf+MIYtqAPmI0tGJGD+CUzWdhL+t0j4bR8XLc9nLyf1sAGvBHmwWfCTKU5Fyg3H2nZO2fQfPj5uRcX6gzv4sKxRZwCUZwPmaS5F0s5CeAfcDPhGf8CXZDM2FOSuSlXmQEHL/8uMtFLiB4K0C0kmYEDs1pLYEbgt0bA9Up5

JyHaWtqarjyyiv57LPPEBGamppa3/c73/kO6enpe33eFVdcwe9+9zvmzJnDu+++y/z58zvoG3cczxKEiASA
R4EzgHLgfRF5SVWDMYc9CMxX1d+KyHTgfuAKoBI4SVUbRKQHsNI9N+RVvC2CoTD5ORn0y8v2+qOM
MR578803Wbx4Me+++y45OTlMmzaNCRMmsHr16rjfI7Z7aNvnCLp37966/O///u+cdtppvPDCC6xfv55p
06Yd8H3nzJnDN7/5TbKzs5k1a1ZrAulKvGykngKsVdV1qtoILADOa3PMaGCJu7y0Zb+qNqg7s9y+M49
1IWCjNmQE/rv21MEqipqSE/P5+cnBxWr17Ne++9R319PcuWLeOLL74AaK1iOuOMM3j00Udbz22pYur
bty+rVq0iGo0esI2gpqaGgQOdtsunnnnqqdfsZZ5zBr3/969aG7JbPGzBgAAMGDODee+9lzpw5HfelO5CXF
96BwKaY9XJ3W6yPgZnu8grogUAIjIIBFZ4b7HTzuj9NAUifLpllob4tuYJDFjxgyam5sZNWoUt956Kyeee
CJFRUXMmzePmTNnMn78eGbPng3AD3/4Q7Zv385xxx3H+PHjWbp0KQAPPPAA55xzDieffDL9+/ff72f9
4Ac/4LbbbmPixI179Wq6+uqrGTx4MOPGjWP8+PH+PE8/fTTrfsuu+wyBg0axKhRozz6BY6MqKo3byzyLWCG
ql7trl8BnKCq18YcMwD4JTAMWAZ8zgcCBynqjvaHPMi8E1V3dLmM+YCcwEGDx48ecOGDUcU8+rNYWb84
m0evngC502wXkZGHIlVq1Z12QtfV3HttdcyceJEvvOd78R9vr3vTEQ+VNVT2a/QffzyVxtfRRdqD6gBzo8Gb
pp3CEaEMDZnRrL5SiWxEZTAbG2RRgvbIWRExEwtRt0bltlHmbBDmi4/mOSzjkazpXM3a9dIKyLRIYwIDpsNLm7TKrK7AcWh4Hvg7QafnMVmeSoR9P70
zRqojTHG035VqvoK8EqbbXfGLC8EFrZz3iJgnJextfOZlIVqOGf8gM78WGOM6bJsLCZXxY46wvXN1v5gjD
EuSxCuMpsDwhhj9mIJwlUWCpMmMLKfJQhjjAFLEK2CoTDi3rI6Gi3rQLTPgdyjGGJ/06GE9GGN1vcE/fBIM1V
AytLffYRiTnP56K2zu4I6I/cbCmQ8c/LgE1Nzc3CXGZrISBLB9VyOhmnprfzAmydx66617ja909913c++993L
66aczadIkxo4dy5/+9Ke43mvnzp37PW/+Q2lcccUVAGzZsoULLriA8ePH378eN555x3Wr1/Pcccd1
3regw8+yN133w3AtGnTuOGGGygpKeHhhx/mz3/+MyeccAITJ07ka1/7Glu2bGmYxKmBmNNY86cOYwdO5Zx48b
x3HPP8cQTT3DDDTe0vu/jjz/OjTfeeNi/WytVTYrX5MmT9XxdbdDfg9zjN6CwaDfIegHH
3ygp556auv6qFGjdOPGjVpTU6OqqlVVVXrUUdpNBpVVdXu3bvv972ampraPW/lypU6YsQIraqqUlXV6
upqVVW96KKL9KGHHlJV1ebmZt2xY4d+8cUXOmbMmNb3/PnPf6533XWXqqpOnTpVv/vd77bu27ZtW
2tcjz/+uN5002qqvqDH/xAr7/++r2Oq62t1eHDh2tjY6Oqqp500km6YsWKfb5De38nQKnu57rqfxmmC7
AhNoxJThMnTmTr1q2EQiGqqqrIz8+nX79+3HjjjjSxbtoy0tDQqKirYsmUL/fr1O+B7qSq33377Pr7vqSq33377PuctWbKE
WbNmUVhYCOyZZ72HJkiWtczwEAgF69ux50EmDWUljY2Pr/BX7m7di+vTpPvzyy4
waNYqmpibGjh17iL/WvixBAGWhGvrlZVPQI8vvvUIwxHWzWrFksXLiQzZs3M3v2bH7H/+99TVVXF8uXLycjI
YOjQofvM89Cewz0vVnp6mL61VUftz9dVX85Of/ISRI0d22PDh
1gYBBCvD1v5gTJKaPXs2CxYsYOHChcyaNYuamhr69OlZEmYGcyotmzZ1NZWUljY2Pr/BX7m7di+vTpvPzyy4
waNYqmpibGjh17iL/WvixBAGWhGvrlZVPQI8vvUIwxHWzWrFksXLiQzZs3M3v2bH7/+99TVVXF8uXLycjI
YOjQofvM89Cewz0vVnp6OtFotHX9QPNLXHfdddx0002ce+65vPnmm61VUftz9dVX85Of/ISRI0d22PDh
1gYBBCvD1v5gTJKaPXs2CxYsYOHChcyaNYuamhr69OlDRkYGS5cuJd5RoPd33vTp03n22Weprq4G9sz3c
Prpp/PYY48BEIlEqKmpoW/fvmzdupXq6moaGhp4+eWXD/h5LfNL/Pa3v23dvr95K03444QQ2bdrE008/zS
WXXBLvz3NAKZ8g6psifF61yxKEMUlqzJgx1NbWMnDgQPr3789ll11GaWkpY8eOZf78+YwcOTKu99vfeWPGj
WPGjOGOO+5g6tSpjB8/npuugmAhx9+mKVLlzJ27FgmT55MMBgkIyODO++++8kylTpnDGGWcc8LPvvvtu
Zs2axeTJk1urr2D/81YAXHTRZxyyilxTZcaD8/mg+hsJSUlWlpaesjnvdU28OOXg1xUMoivjCg8+AnGmLjY
fBCd75xzzuHG8k/NNPb3d/V5oPIiEU5WbxyCUTLTkYYxLWjh07OOaYY+jWrdt+k8PhENdr+k8PhsEZqY4yJ8cknn7
Q+y9AiKyuLv//97z5FdHC9evevVizZo1Hf6+liCMMZ5RVUQSa36VsWPH8tFHH/kdRoc7nOaEoaElK9iMsZ4Izs7
m+rq6sO6MJmOpapUV1eTnZ19SOdZCIy44ni4mLKy8Ly8sPSk3CLi4sP6RxAeBgLAb1T1g
Tb7h+DMQ10EbAMuV9VyEZkAPAbkARHgPlX9o5exGmM6VkZGGRuvTvyYxeVbFJCIB4FHgTGA0cImIjG5z
2IPAfFUdB9wD3O9u3w38SzA6sGxoYIm7vLRlv6quU
dXP3OUQsBWnlGGMMaaTeJkgBgKbYtbL3W2xW3qfXbibg0H2BMsBzzc8gw8Ms0SHlx8lYpr2PyF/IipiifiiF
wFLAMqcNocABCR/sD/AleqarTtySIyF5jrru4Ukxn4/Drrdqd2FbFP7AT4MlnD2dTz2ZxSMsu3ZVHbm0b5fiblb
TAoZr3Y3dbKrT6aCSAiPYALVXWHu54H/AW4Q1Xfa+8DVHUeMK8jghWRR0v09bp5q7LfYm/0ee7PfY49k
/y28rGJ6HxghIsNEJBO4GHgp9gARKRlhhuw+eRRHv8CzgN2As9jNEYY8x+eJYgVLUZFzuBZ4DVgFPKOqK0TkRERkrltllfLst9ib/R57s99jj2T/LSxBG
GOMaZeNxWSMMaZdliCMMca0K+UThIFPRWStiNzqdx+EpBIrJURIiUiUiYi1/sdk99PRNpNpQkBdxIvDNFpZJmBcDdUm5xH9RLdVEAjtAq9PkFPPt1r5j/S6/V/P1/s2Hp7YVWCD4F2hoUlzJ6n6xq8QSAkWE
hEi0ktEForlahFZJSln+R2Tn0TkRvf/yUoR+YOIHNpQzQkgpRNEnONFpZJm4F9VdTRwIvD9FP89AK7H6H6YV
nnIE3X1XVkcB4Uvh3EZGBwL8AJap6HM6ApBf7G1XHS+kEQXzjRaUMVa1U1Q/c5VqcC0Db4VFShogUA
2cDv/E7Fr+JSE+cruf/A6CqjS0PtaawdKCbiKQDOUDI53g6XKoniHjGi0pJIIjIUmAh03XkWvfcL4AfAPsO8
BhQBXwpFvl9hsR6e53UH5R1Qqc0ag3ApVAjaq+7m9UHS/VE4RphzvsyxsPXADaoa9jseP4j4AcBWVV3udy
xdRDowCXhMVScCu4CUfCYoIvzgxH6vGu4hc7m9UHS/VE8RBx4tKNSKSgZMcfq+qz/sdj49OAc4VkfU41Y9TRdK5O5KSp8HFcfq+qz/sdj49OAc4V

```
kfU4VY/TReR3/obkq3KgXFVbSpQLcRJGqvoa8IWqVqlqE/A8cLLPMXW4VE8QBx0vKpWIM7v8/wCrVPU
//Y7HT6p6m6oWq+pQnH8XS1Q16e4Q46Wqm4FNInKsu+l0IOhjSH7bCJwoIjnu/5vTScJGe7+H+/aVqjaL
SMt4UQHgCVUt8zksP50CXAF8IiIfudtuV9VXfIzJdB3XAb93b6bWAXN8jsc3qvp3EVkIfIDT++9DOmhk6a7
EhtowxhjTrlSvYjLGGLMfliCMMca0yxKEMcaYdlmCMMYY0y5LEMYYY9plCcKYgxCRSMzUtx915Ki/IjJURF
Z21PsZ05FS+jkIY+JUp6o2J7pJOVaCMOYwich6EfmZiHwiIv8QkaPd7UNFZkLu9r4i8i8IClfu6+
WoRkCIvK4O7fA6yLSzT3+X9y5OVaIyAKfvqZJYZYgjDm4bm2qmGbH7KtR1bHAL3FGfwX4L+C3qjoO+D3
wiLv9EeAtVR2PM45Ry1P7I4BHVXUMsAO40N1+KzDRfZ/vePXljNkfe5LamIMQkZ2q2qOd7d7euB6aq6zh3
kcLOqFojIl0B/VW1yt1eqaqGIVAHFqtoQ8x5DgUWqOsJdvwXIUNV7ReRVYCfwIvCiqu70+KsasxfrQRhzZ
HQ/y4eiIWY5wp62wbNxZjycBLzvTkxjTKexBGHMkZd8e77vI77Jl+8jLgbXf5DeC70DrXdc/9vamIpAGD
VHUpcAvQE9inFGOMl+yOxJiD6xYzui048zK3dHXNF5EVOKWAS9xt1+HMvPZvOLOwtYx6ej0wT0T+Caek
8F2c2cjaEwB+5yYRAR6xKT5NZ7M2CGMOk9sGUaKqX/odizFesComY4wx7bIShDHGmHZZCcIYY0y7LEE
YY4xplyUIY4wx7bIEYYwxpl2WIIwxxrTr/wP4xg0HT2dSPgAAAABJRU5ErkJggg==\n"
```

```
        },
        "metadata": {
         "needs_background": "light"
        }
       },
       {
        "output_type": "display_data",
        "data": {
         "text/plain": [
          "<Figure size 432x288 with 1 Axes>"
         ],
         "image/png":
```

```
"iVBORw0KGgoAAAANSUhEUgAAAYgAAAEGCAYAAAB/+QKOAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA
6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAgAElEQVR4nO3de3zcdZ3v8ddncm2TTHpLm0wv9EKhnbRStC
AcF1DkUm90XVfLRRc8Kg9ZEW/LEQUVWTzeztGze+SoHBdXXVjoAut2D0hFQZFd0Zba0qaFUkopSW/p
Nb3mNp/zx++XZJJO2qTNL79M8n4+HvOY+V3nM9Pm957f7/v9/X7m7oiIiPSUiLsAEREZmhQQIiKSkwJC
RERyUkCIiEhOCggREcmpMO4CBsqECRN8+vTpcZchIpJXnn/++d3uXpVr2rAJiyxARyStm9lMTa2kp9fT3cmRQQIpLX6uvrqaioYP
r06ZhZ3OUMSe7Onj17tL+fPKyxY8pHpDDtgDP9Wvm7X73M2voDcZciIjJKkjPGiEqBvLtJsH3kcMVyNtXv4OYmLuoODNuVGy0cqXt2jPHjxyscTsDMGD9+fL/3biCzEH1NZ7b/ZxuP5+pOWXwcoYekOz+a2jXb9BbkSAmh9svcHQYeYXDcIkhLgAmNkiM3vJzbPXiTDFJzOfVzwFunHfHb1FA3hxDiP8a2Eze4RB3mVvoEVUHA
nTZtbWNJOu3KSBE5PS4O7feeivz5
s1j/vz5PPTQQwBs376diy++mAULFjBv3jx+97vf0d7ezg033NA573e/+92Yqz9eZN1czawAuAe4HKgHVpj
ZMndfnzXbA+7+g3D+q4DvCoLgaqAVSwK/M7Cx3b4+i1nRNksfX7uDgsVYqSvveR1hEhpav/nvdgP/Y
S6eSfOU9tX2a99FHH2X16tWsbOG3bt3c95553HxxRfzL77/PFWa7NGkn+NQ0N
Daxbtw6A/fv3D2jdAyHKPYYjzgU3uvtndW4AHgcXBg+zFwIPu4KNKvCmOvV5o0xm9DAcXZM7h79r9kGdBxg+zFwIPu3uzurwKbwvVFIp1KAvDijo
NRvYWIjADPPvss11xzDQUFBUyaNIlLLLmEFSt/9HjH/+YO++8k7J3kJ3ni
iSdIJpNxl3+cKE+Umwy8njjVcD7y550xm9gngs0AxcGnWBKCu4QD
nTR93yusRkXj19Zf+YLv44ot55plneOyxx7jhhhv47Gc/y/XXX8+aNWtYvnw5P3LiVVRKTRdpxcKwo0tE4tf9x6tesLnIlZ92/c0jP+3vU3Hh3kHxO6JH1m7HVMjKcf/DR4tbHYdN2nF7ebMSnWY4MlFEoTIDrZJSuJvfwKSKj7EfJGi/Xu74HsZYmwnPmSjOoqrz90mh7Mfnb7w9gvTgNMTF+xk8fQKCU4JGYYmVJ1jUtoW/aJkslqs0U3ZkJ6+N5VH5pHb/y3hHRKr4f2b9/1t7OCkSmZ9rtkzV8sHYLTE+PBZJhfXrYuaJdwyv9GBZiiSL80lMh0WYPeeNvXPTqXSJgHxmdkdcwq3lLwkD1b1Rzs78JhNkBvJQqTxP5BA49P9SXLEJCfpE
1/1BAFBI0xYVR7FZ8pIoJqXO1DXWiDfHBE6u9FsuC7Ewdvv/lP4lHnDF+0NZL6AHwvzJaKKVIKi8CjE5uhXaaWeryFdEQl47i+Dk5P4GQASIg4gSZKFNlVdE7CYwAYPm4hnwtjZJoOPWf36JNhkvwEBiD6GDGnjoCvlA0RmaiHsgGCfxAuT42C0jtuNmdXMSa8O5Uv/HFwMWp5mEvggK4Ws5c9PQtA1hCk6pb5u+gdDmADlTpe3dFgiVxkmZB5PcawHF4CvfvrCy3AatwVJk3ew/cPhr9FbHAaZYyqV+FwaQPuVZ/+rlPw3PnR05AWmuAwqwjHTE2ypk56RtH9GE0c7A
3t5Dt6aUZsp6BWTW30zq8d8K5kQ5mHchNjC4Dm03d+4HbsOBKS3OKbt2c5
/Q9SSMWzibFbHd3o3dvstMv0h5atXefHF9FM+nYQ6RGBSk89Mdr9RKFOsNXLq4uYwcrPOxmh3rLNazy+62GAKjF/r1xK9L5WmDsdDAiPwPv
nTR93yusRkXj19Zf+YLv44ot55plneOyxx7jhhhv47Gc/y1/91V+xZs0aIi9fzg9+8AOWLl3KffdZFe3p3cTeSO
```

3u97j7LODzwB39XPZed1/o7gurqnJezrxPJlaUML6sWA3VInJaLrroIh566CHa29tpbGzkmWee4fzzz+e11
15j0qRJfOxjH+OjH/0oq1atYvfu3WQyGd73vvdx9913s2rVqrjLP06UexANwNSs4SnhuN48CHz/FJc9LWZ
GOpVUQIjIaXnve9/L73//e8455xzMjG9961tUV1fzk5/8hG9/+9sUFRVRXl7OT3/6UxoaGvjwhz9MJpMB4
Otf/3rM1R/P3P3kc53Kis0KgY3A2wk27iuAa929Lmue2e7+cvj6PcBX3H2hmdUCDxC0O6SAXwOzT9RIvX
DhQj+dGwZ9/Rcb+PGzW6i760qKCmLfsRKRPtqwYQNz586Nu4y8kOu7MrPn3X1hrvkj24Nw9zYzuxlYDh
QA97l7nZndBax092XAzWZ2GdAK7AOuD5etM7OfAAuD5etM7OlwHqgDfhEVD2YOqRrkrS0Z9i06xBza4ZeY5GIyGCL9G
qu7v448HiPcV/Oev2pEyz7NeBr0VXXXWdD9bYmBYSICEOgkXqomDGhjFFFBTphTkQkpIAIFSSMOTUVr
N+ui/aJiIACopt0TXDJjaga7kVE8okCIks6laTpWBv1+47GXYqISOwUEFmyG6pFREY6BUSWsydVkDB0wpy
IROZE947YsmUL8+bNG8RqTkwBkWVUcQGzqsrVk0lEhIjPg8hH6VSSFa/ujbsMETkVv7gNdqwd2HVWz4
d3fKPXybfddhtTp07lE5/4BAB33nknqZSPAb33nknhYWFPP300+zbt4/W1lbuvvuFi9e3Os6cjjl27Bg33XQTK1eupLCwk
O985zu87W1vo66ujg9/+MO0tLSQyWR45JFHSKSOADH6C+vp729na+9KUvsWTJktP62KCAOE66Jsm/
rd7GvsMtjC0rjrscERnilixZwqc//enOgFi6dCnLy/nlltuIZlMsnv3bi644AIuuuoqzKzP673nnnswM9auXcuL
L77IFVdcwcaNG/nBD37nBD37Apz71Ka677jpaWlpob2/n8ccfJ5VK8dhjjwFw4MDAdNdXQPTQ0VC9fnsTbzlzQ
szViEi/nOCXflTOPfdcdu3axbZt22hsbGTs2LFUV1fzmc98hmeeeYZEIkFDQwM7d+6kurq6z+t99tln+eQnP
wnAnDlzOOMM9i4cSMXXnghX/va/16ivr+cv/uIvmD0D17NvPnz+dzn/scn//853n3u9/NRRddNCCfTW0QP
XTcPEjtECLSV+9///t5+t5+OGHeeihh1iyZAn3338/jY2Nyx7NixAXmva6/6+9lmXLjFq1Cje+c53
8tRTT3HWWexatUq5s+fzx133MFdd901901IO+lPYgexpUVU1NZqp5MItJnS5Ys4WMf+xi7d+/mt7/9/LUu
LmXixIkUFRXx9NNP89prr/V7nRdddBH339/8dddBH3338/l156KRs3bmTr1q2cffbZbN68mZkZZ3LLLbewLLLbewdetWXnjhBe
bMmcO4ceP44Ac/yJgxY/jRj340IJ9LAZFDuiZJ3TZdckNE+qa2/PjWxyMmxNMg//W/Pnzz
bhwIXPmzOn3Ov6r/+am266ifnz51NYWMg//uM/UlJSwtKlS/nZz35GUVER1dXXVfPGLX2TFihXceuutentJBIJ
ioqK+P73v3/yN+iDyO4HMdhO934Q2f7nL1/i//zmZF/eqMnPHFetNogcqhNJWnPO
C/tOBh3KSIisdEhphzSNV09mc6ZOibmakRkuFm7di0f+tCHuo0rLS3lueeei6miXDKxFUlBSqJ5
NInnD3fp1jELf58+ezevXqQX3PU2lO0CGmHMyMuSk1Vlvkg9LSUvbs2aPL9J+Au7nnzx5KS0v7tZz2IHqR
rkmydOXrtGecxFm7di0f+tCHuo0rLS3lueeei6miXDKxFUlBSqJ5
qKipixowZcZcxLOkQUy90RrWIjHQKiF7MnlhBUYHpjGoRGbEUEL0oLkwwe2KF7i4nIiNWpAFhZovM7CU
z22Rmt+WY/lkzW29mL5jZr83sjKxp7Wa2Onwsi7LO3qRTSR1iEpERK7KAMLMC4B7gHUAauMbM0j1m0dj
xOw0N3fADwMfCtr2lF3XxA+roqqzhOpTSXfaiZXU0DcxVGEZF8EuUexPnAJnff7O4twIPAldkzuPvT7
HHwO6F8frIila4KjG6jq1Q4jICBRlQEwGXDs8arg/H9xmRmZ5thZsKA1UC33khm
di7wQ4Jw2JU1fqyZlYSvJwBvAdZHWGuvalNJXt19mEPNbXG19mEPNbXG8vYhIbCILCHdvA24GlgMbgKXuXmdmd5lZ
R6+kbwPlwL/06M46F1hpZmuAp4FvuHssAdFxRvVL2rgtIiNQpFdzdffHgcd7jPty1uvLeixugyCgIhCfTY
VZsK7g1Rt62JhdPHxVyNiMjg0ZnUJzEpWcrosiJtqBaREUkBcQJmRromyQs1WsolIiKOAOAkzIxJjFWLWMiMBWiK2tKNWaQ9b1ZRmLwVQkDXuUOvnBgHM0fMzWYpaFmzXO
V/ZsK7g1Rt62JhdPHxVyNiMjg0ZnUJzEpWcK4smI1VIvIiKOAOAkzU0O1iIxICog+qE0leWnHQVrbM3GXIiI
yaBQQfZBOJWlpz/BK46G4SxERGTQKiD6oDXsy1TXoMJOIjBwKiD6YMaGc0qKE2iFEZERRQPRBC4u1q
X/haRkUUB0Ue1qSR12w7gPuCXfBIRGZIUEH2UrknSdKyK2Nhv1H4y5FRGRQKCD6qLOhWoeZRGSEUED0
0ZzqJAnTvSFEZORQQPTRqOICZkwbawUEJuq1B6EiIwYCoh+SKeSNOw/yv4jLXGXIiISOQVEP
9TqHtUiMoIoIPphbk0YEGqHEJERQAHRDxPKPKS5iULNEehIiMCAqIfqpNVepcCBEZERQQ/ZSuSbKp8RDHW
tvjLkVEJFIKiH6qTSVpzzgbdx6Mux5MIjJCKCD6aerY0ZSXFKonk4gMewqIfkokjHjHRNUg3VIj
LsRRoQZrbIzF4ys01mdluO6Z81s/Vm9oKZ/drMzsiadr2ZvRw+ro+yzv5Kp5Js2N5EJqN7Q4jI8BVZQJhZAX
AP8A4gDVxjZukes/0JWOjubwAeBr4VLjsO+ArwzZuB84CtmNjaqWvsrnUpypKWdLXsOx12KiEhkotyDOB/
Y5O6b3b0FeBBYnD2Duz/t7kfCweAKeHrK4En3X2vu+F4PUVBsRk4PWs4fpwXG
8+AvziFJcdVLMnIVOYMPVkEpFhrTDuAgDM7IPAQuCSfi53I3AjwLRp0yKoLLeSwgTmT6pQQ7WIDGtR7k
E0AFOzhqeE47oxs8uA24Gr3L25P8P2RrknqEJOIDGtRBsQKYLaZzTCzYBqYFn2
DGZ2LvBBgnDYlTVpOXCFmY0NG6evCEcNG6evCMcNGbWpJI0Hm9l18FjcpYiIRCKygHD3KygHD3NuBmgg37BmCpu9eZ2V
1mdlU427eBcuBfzGy1mS0Ll90L/C1ByKwA7grHDRk6o1pEhrtI2yDc/XHg8R7jvpz1+rITLHsfcF901Z2ezo
DY3sRbz54YczUiIgNPZ1KfomRpEVPHjVJDtYgMWwqI05CuSUpip5dc9hDje3xV2KiMiA
U0CchnRNEnd4cYf2IkRk+FFAnIbayUFDtdohRGQ4UkCchupkKWNHF6mrq4gMSwqI02BmpFM6o1pEhi
cFxGmqTVXy4o6DtLZn4i5FRGRAKSBOU7omSUt7hs2Nh+MuRURkQCkgTlNnQ7V6MonIMKOAOE0zJ5RRXJhQQ7WIDSwFxmmZMKK

OkMKGGahEZdhQQp6mwIMGc6go1VIvIsKOAGADpVCV125pw97hLEREZMAqIAZBOJTlwtJVtB3RvCBE
ZPhQQA6CzobpBDdUiMnwoIAbAnOoKzFA7hIgMKwqIATC6uJAZE8rUk0lEhhUFxACpDRuqRUSGiz4FhJl
9ysySFvgHM1tlZldEXVw+Sdckadh/lANHWuMuRURkQPR1D+K/unsTcAUwFvgQ8I3IqspDnQ3V29VQLS
LDQ18DwsLndwI/c/e6rHECzK0JAkLtEClyXPQ1IJ43s18SBMRyM6sAdPnSLFUVJUysKFFPJhEZNgr7ON9H
gAXAZnc/YmbjgA9HV1Z+qk0ltQchIsNGX/cgLgRecvf9ZvZB4A5AB9t7SKeSbNp1iGOt7XGXIiJy2voaEN8H
jpjZOcDngFeAn55sITNbZGYvmdkmM7stx/SLwx5RbWb2lz2mtZvZ6vCxrI91xqo2VUlbxnl556G4SxERO
W19DYg2D65Etxj4nrvfA1ScaAEzKwDuAd4BpIFrzCzdY7atwA3AAzlWcdTdF4SPq/pYZ6zSHQ3V6skkIsN
AX9sgDprZFwi6t15kZgmg6CTLnA9scvfNAGb2IEHArO+Ywd23hNOGRYP3tHGjKS8pVDuEiAwLfd2DWAI
0E5wPsQOYAnz7JMtMBl7PGq4Px/VVqZmtNLPnzOzPc81gZjeG86xsbGzsx6qjkUgYc2sqdEa1iAwLfQqI
MBTuByrN7N3AMXc/aRvEaTrD3RcC1wL/y8xm5ajrXndf6O4Lq6qqIi6nb9I1STZsbyKT0b0hRCS/9fVSGx
8A/gi8H/gA8Ieejco5NABTs4anhOP6xN0bwufNwG+Ac/u6bJxqU5Ucbmnntb1H4i5FROS09PUQ0+3Aee
5+vbv/FUH7wpdOsswKYLaZzTCzYuBqoE+9kcxsrJmVhK8nAG8hq+1iKEundEa1iAwPfQ2IhLvvyhrec7Jl3
b0NuBlYDmwAlrp7nZndZWZXAZjZeWZWT7Bn8kMzqwsXnwusNLM1wNPAN9w9LwJi9qRyChOmnkwi
kvf62ovpCTNbDvxzOLwEePxkC7n74z3nc/cvZ71eQXDoqedy/wnM72NtQ0pJYQFnTixXQ7WI5L0+BYS7
32pm7yM41ANwr7v/a3Rl5bd0KsmzL++OuwwRkdPS1z0I3P0R4JEIaxk2alOVPLqqgcaDzVRVlMRdjojIKT
lhQJjZQSBXf00D3N2TkVSV57rOqG7ikoqh0f1WRKS/TtbQXOHuyRyPCoVD79K6N4SIDAO6J3UEKkcXM
WXsKOq2qSeTiOQvBURE0jVJ3TxIRPKaAiIitalKXt19mMPNbXGXIiJyShQQEUmnkrjDizsOxl2KiMgpUUBE
pPOSGzrMJCJ5SgERkVRlKWNGF7FeDdUikqcUEBExs6ChWl1dRSRPKSAiVJtK8uKOg7S1D4sb5onICKOAi
FA6laS5LcPm3YfjLkVEpN8UEBFK11QC6IQ5EclLCogIzaoqo7gwoXYIEclLCogIFRYkmFNdoa6uIpKXFBARq
00lqdvWhHuui+KKiAxdCoiIpWuS7D/SyvYDx+IuRUSkXxQQEes4o1q3IBWRfKOAiNic6iRmujeEiOQfBUTE
ykoKmTG+jPXb1dVVRPKLAmIQpMOGahGRfKKAGATpVJL6fUc5cLQ17lJERPos0oAws0Vm9pKZbTKz23J
Mv9jMVplZm5n9ZY9p15vZy+Hj+ijrjJruUS0i+SiygDCzAuAe4B1AGrjGzNI9ZtsK3AA80GPZcD5wN
fMbOxUdUatdpUcMkNnTAnIvkkyj2I84FN7r7Z3VuAB4HF2TO4+xZ3fwHoebnTK4En3X2vu+8DngQWR
VhrpKoqSqiqKNEehIjklSgDYjLwetZwfThuwDDcDcEa1ejKJSP7I60Zqd7/X3Re6+
8Kqqqq4yzmhdE2STbsO0dzWHncpIiJ9EmVANABTs4anhOOiXnZISqeStGWcl3ceirsUEZE+iTIgVgCzzWy
GmRUDVwPL+rjscuAKMxsbNk5fEY7LW50N1WqHEJE8EVlAuHsbcDPBhn0DsNTd68zsLjO7CsDMzjOzeu
D9wA/NrC5C5cdi/wtwQhswK4KxyXt84YN5qy4gL1ZBKRvFEY5crd/XHg8R7jvpvz1egXB4aNcy94H3BdlfYMp
kTDm1qihWkTyR143UuebdCrJhxu0HyWR0bwgRGfoUEIMoXZPkUHMb9fuOxl2KiMhJKSAGUW2NLn0hIvlDATGI
zqkqJ2FQ161rT+5SwgoYIJiG8dGKPsh5wCWpC+NGV3M5DGjdI9qERnyFBAxSKeSehaRiOSHEZNhbWH
ceibsUEZGTUkAMIp1RLSL5R
AExiGZPKqcgYerJJCJ5QQExiEqLCpg9sVwN1SKSFxQQgyxdk9QhJhHJCwqIQZZOJdnZ1MzuQ81xlyIickIKiE
GWTuneECKSHxQQg6zz5kE6zCQiQ5wCYpCNGV3M5DGjdI9qERnyFBAxSKeSrFdPJhEZ4hQQMahNJdm
8+zBHWtriLkVEpFcKiBika5K4w4s7DsZdiohIrxQQMVBPJhHJBwqIGEweM4rKUUVqqD6RtmY4pnYakThF
ej8Iyc3MdEZ1bzIZeOEh+PVdcGgnnHkZLLgGznoHFJXGXZ3I0NF6FPa+CntfAUvAnHcN+FsoIGJSm0rys+d
eo609Q2GBduQA2PIfsPyLsH01pM6F+e+DtY/AvyyH0jEw732w4DqY/EYwi7takei1NcO+LbDnlSAIOp83
Q1MDEN5bpnq+AmI4SaeSNLdleHX3YWZPqoi7nHjt3QxPfhk2/DskJ8N774+X574dEAi77Krz6W1j9AKy+H
1b+A0w4C865Bs65GpKpuKsXOT3trbB/a44ZGvjZsH0twTP42fBuJnBIwIKiJh0NlRvbxq
5AXF0PzzzbfjDD6GgCN52O1x4MxSP7ponUQCzLg0exw5A3c9hzT/Dr78KT/0tzHxrsFcx511QNCquTzK8
HdwJ9SugYSVsfwHaW7qmmQHW++vOPb1cr3tbjpMsByQKoSQJpZUwakzwXFoZ7Gl2jhsTzFMwBDZzm
fYgBDp+/WcHwb7XwNu75i1JBhv+KecHP4Syg2D0uEEtewh8cyPTrKpyigsT1G1rYvGCyXGXM7ja2+D5H
8PT/x2O7gs28JfeAcmaEy9XWglvuj547HkF1jwYPB75SPBHVfvncM61MO0CHYI6VW3NQQjUrwgfK+HA
1mBaoggmpaG4HNwBD5/p5XU4nOt1t3np/3KZNmhuCn5kZG9ccymu6D1Iuo3PMa64vO//lzIZaKrvfhio
Iwj2bYFMa9e8RWUwfiZUvwFq35sVArOgbMKQ+f+rgIhJUUGCsydVjKyuru7w8pPwyztg90sw/SK48mtQ
c07/1zV+Flx6O7z1C/Das7D6n4P2ilU/DX5pdRyCGjNt4D/HcOEO+18LQqAjELa/0LUhq5wKUxbCBR+HKe
cFG7Oh1lHAHVoOB3uXx/YHz0fD597G7d8Kx9YG41pOci6SFeQIkqww8UxXQ/HeV6E96yrNhaOC/4sT5
wR7uB0BMH4WlE8aMiFwIuadSZ7fFi5c6CtXroy7jH657ZEXWF63g1VfuhzLg/8sp2VnHSy/HTY/HfzRXHE
3nP3Ogf0jaT4EG5YF7RVbfheMm34RLLgW5l4FJeUD9175qPkgNKzq2jNoWAmHG4NpRaMh9cYgEKacF
zxXVMdb72BoD/dEThQu3cb3GAcwbka44Z/ZfU+goiZoRxvizOx5d1+Ya5r2IABajnQ/7j1I0qkkD654nR1N
x6ipHKbHzw/tgqe/FvyyL6mAK78O530UCosH/r1KyoMwWHBtcFz3hYeCsPj5TfDY30B6cdBl9ow/y4s/3
NOSyQR7aZ17Byth13o6D9lMOAvOvLwrECamh8ax+sFWUBgc1z/VY/vuebEncKoi/R9hZouAvwMKgB+5

+zd6TC8Bfgq8CdgDLHH3LWY2HdgAvBTO+py7fzySIo81wTenB6lfsyA43JFaEHQbK62M5C07dF76e1vT8
AuI1mPw3P+B330H2o7C+TfCJZ8fvEa2sWfAJf8NLr4Vtj4Hax6Adf8aPFdOCw4/Lbgmst4fg+7wnmCPoO
NQUcOq4JcxBIdDpiyE9FXB8+Q3Bb1h5PQN43CACAPCzAqAe4DLgXpghZktc/f1WbN9BNjn7mea2dXAN
4El4bRX3H1BVPV1yrQFG5Lta2DLs7B2ade0cbO6AqPmnOAxgH9Yc2qSmEHdtibePnfSgK03Vu5Q9yg8e
WfQuHn2O+Hyu2DC7HjqMYMzLgwei74JLz4WhMQz34ZnvgXTLgzaK2rfC6XJeGrsr7YW2Lmue9vBvleD
aVYAk2qDbsIdewfjZg3/PSaJRJR7EOcDm9x9M4CZPQgsBrIDYjFwZ/j6YeB7NtgH40ePg7fe1jV8aFcQFttX
w7bVwR9h3aNd08eckRUYC4JH2fhTeuvykkKmjy8bPg3V9SvhiS9A/R9h0nxY/G9BN9Shong0vOH9weN
AQ9chqH+/BX7xeZj77uDw1IxLgu61g809ODu25VDQXtDcFLSrNB8Mjnl3hML21dB2LFimvDoIgjfdEIRBa
gEUlw1+7TIsRRkQk4HXs4brgTf3No+7t5nZAaBjazvDzP4ENAF3uPvver6Bmd0I3AgwbdoA9VYpnwizLw8
eHQ7vgR1rgsDoCI/1/9Y1vXJqVmCEexzlE/v0dulUkhfq9w9M7XHZvxV+9VVY9zCUTYSr/nfQdTWOjWxf
VU6Giz4Lf/YZaHg+CIp1D8Paf4GKFJyzJOgyW3XWydfV1hJsxFsOhhv2g+GGvSkcfyhrfNYj1/gTddksKAn+
b5330a69g+TkYX+YQ+IzVFultgPT3H2Pmb0J+LmZ1bp7t5/a7n4vcC8EvZgiq6ZsfNfJWh2O7gu6BHYExvY
18OL/65peUdM9MGrOCcb1+GNO1yR57IXtHDjaSuWoosg+QiSaD8Kz34Xf3xMMX/Q38GefDhqj84VZu
LFdCFf+d9j4iyAs/uPvg882eSFUz+v6JZ8rCLK7Np5IcXnw3ZRUdL0uq+oalz2+JBk0uneOrwi67EbRuC/Siyg
DogGYmjU8JRyXa556MysEKoE9HvS9bQZw9+fN7BXgLGDo9GMdNRZmXhI8Ohxrgh1ruwJj22rY+ASdP
UfKJvZo01hAuibYmG7Y3sQFM0/tUNWgy7TDn/4JnrobDu+C+R+At38Zxkw9+bJDWVFp0BZR+97g7OG1
S2HNQ0G7RfaGOjklHO6xAe/c0JeHG/isDX5xudoBJO9Edh5EuMHfCLydIAhWANe6e13WPJ8A5rv7x8N
G6r9w9w+YWRWw193bzWwm8Ltwvr29vd+QPQ+i5XAYGlmHqBpf7DyUkBk1nmcPpaicdR7nLLwo6FU
zZloQQEPx0MHm3wTnM+xcB1PfHPzqnpKzC7WI5IFYzoMI2xRuBpYTdHO9z93rzOwuYKW7LwP+AfiZm
W0C9gJXh4tfDNxlZq1ABvj4icJhSCsuCy79MO2CrnGtR4MTx7b9icT2NUz607OcueUnsOW+rOXCQwq9P
QY7QBo3wpNfCvaIxkyD9/8jpP98aIaYiAwInUk9BFx/3x/Z13SIZUsmBJc+2L+1+2Pfa8dfEmCwAuTIXvjNN
4KrqBaOgos/B2++aehdckFETonOpB7i0qkkP3plNy1V8yiuecPxM7gHp/j3DI6Ox5ZnBz5A2lpgxf+F334zaI
x90w3w1i9CedWAfnYRGboUEENAuiZJa7uzcedB5k3Ocfa2WbBBHzU294XtBjpA9m+FX30luE/DrEvhiq8
FV/EUkRFFATEE1GbdGyJnQJxMFAFSNQeuewRmX3YKn0hEhgMFxBAwfXwZo4sLojujur8B4hk4+10j8+J
tItJJW4AhIJEw5tYk47vkxskCRERGJCiR/0sAAAuhSURBVJ25M0Ska5Ks395EJjM8epWJSP5TQAwR6VSSQ
81tPPDHrRw40nryBUREIqZDTEPEpXMmMm3caO74+TruXFbHhbPGs2heNZenJzGxQucciMjg04lyQ0g
m46yp38/yup08sW47W/YcwQwWnjGWK2urubK2mqnjBv/OdyIyfJ3oRDkFxBDl7mzceYgn1u3gibodbN
geNGDXppIsqq1m0bxqzpxYPvzvZS0ikVJADAOv7TnM8rodLK/byfOv7QNgZlVZZ1jMn1ypsBCRflNADDM
7m47xy/U7Wb5uB7/fvIf2jJOqLOWKMCzOmz6OgoTCQkROTgExjO0/0sKvNuziiXU7eOblRlraMowvK+b
y9CSunFfNf5k1npLCIXxnNxGJlQJihDjc3MZvXmpked0OnnpxF4ea26goKeTSuRNZVFvNJWdXMbpYHddE
pIsCYgRqbmvnPzft4Yl1O3hyw072Hm6hpDDBxWdVsai2msvmTqJydJ7d4lREBpwCYoRra8+wYss+ltft4Il
1O9jRdIzChHHhrPFcWVvNFbU610JkpFJASKdMxnmh4UBnWLy6+zBm8KZpY1k0T+daiIw0CgjJyd15eVd
4rsW6HawPz7WYPbGcs6ormDmhjBnhY2ZVOZWjdEhKZLhRQEifbN1zhOV1QdfZzY2HeH3fUdqzLh44vq
y4MzBmVJUxc0I5M6vKmDZuNKVF6iklko8UEHJKWtoyvL7vCK82Hmbz7kO8uvswmxsP8+ruw+w62Nw
5nxlMHjOKmVXl3fY6ZkwoY/KYUSR0TobIkKV7UsspKS5MMKuqnFlV5cCkbtMONbd1C46O8Hj4tX0cam
7rto4Z47P3OsqYWVXGjAnljB1dpLO/RYYwBYSckvKSQuZPqWT+lO63SHV3Gg81h+HRFRwv7zrIr1/cSWt
71x5r5aiiMCzKwj2P8s49j1HFfTtk5e60tjut7Rla2zO0tGeC4bYew+0ZWtu6htt6TmvP0NKWOX5dbU5JUY
Jxo4sZW1bMuLIixo4uZlxZMFxRUqiQk2FLASEDysyYWFHKxIpS3jxzfLdpbe0Z6vcdDUJj92Fe3X2IzY2H+f0
re3h0VUO3eVOVpUyoKOm2wQ428FnD4QY+KsWFCYoSRnNbhrZebuRUmLAgOEYXMzYMj67hHoESPo
8uLlCoSF6INCDMbBHwd0AB8CN3/0aP6SXAT4E3AXuAJe6+JZz2BeAjQDtwi7svj7iV5hQYLpE8qYPqG
Mt/WYdqSljS27jwSHrMJ2jr1HWiguSFBUkKCowILnwkQ4LhwuSAQb8qzh7q8TFBf2GC5IUBSO61h/YUH
2sFGQsM6NuLtzsLmNfYdb2Hu4hX1HWth7uJV94etguIV9h1t5edehzvG93RywuLBrj2Ts6KLugdIxnBUo4
8qK1QlAYhFZQJhZAXAPcDlQD6wws2Xuvj5rto8A+9z9TDO7GvgmsMTM0sDVQC2QAn5lZme5e3tU9Uq
8RhcXkk4lSaeScZdyHDMjWVpEsrSIM8aX9WmZTMZpOtbaPVCOtAQh0/EcjtuwrYm9R1rYf4I7CZYWJSg
tKqAw0RVgRQUJChNGYeezUZRIUNDxuiARzmcUJrrm6Zw/a12FBcHeUkG4jsICO27dHfODYQYWfjfBc/g
gmHDctKxl6DFslv2a3tcfjqfnOjvG0XM9x68rfPvj1te5zvD9jnuPrtUdV3vCjn9OhJ8pYRa8zlp/PolyD+J8YJO
7bwYwsweBxUB2QCwG7gxfPwx8z4JvcDHwoLs3AxvNrCHs70Uzq2vFBdpzGc3MtQ1sYmq2B1CPFTWO
uvgdLc1tV+0p5xWjPB67Zuz05bJsPRVg/mCcd3vm73YL5MpnPejnESva7QyAoVeoRKwjrnIWta5zKJ45eZ

```
W5Pke9e+ccDrjTIgJgOvZw3XA2/ubR53bzOzA8D4cPxzPZad3PMNzOxG4EaAadOmDVjhInEoLEgwvryE8
eUlg/7e7n6SIAlCqLXdcZyO3vHudA57uB4Px9NtfPdpTjDBT7AOssefYP3hrN3W0zXOO+sMl+i2PnK8X9fn
yqo1axwd8xPsKTqQCT9fxoP1Z5zwtWe9pttwX5bJhIVlMscvkwnrybgzLaKrH+R1I7W73wvcC8F5EDGXI5
K3zCxsu0HtHdIpEeG6G4CpWcNTwnE55zGzQqCSoLG6L8uKiEiEogyIFcBsM5thZsUEjc7LesyzDLg+fP2Xw
FMe7BMuA642sxIzmwHMBv4YYa0iltJDZIeYwjaFm4HlBN1c73P3OjO7C1jp7suAfwB+FjZC7yUIEcL5lhI0
aLcBn1APJhGRwaVrMYmIjGAnuhZTlIeYREQkjykgREQkJwWEiIjkpIAQEZGchk0jtZk1Aq+dxiomALsHqJx8
p++iO30f3en76DIcvosz3L0q2Sxim8xW9taSP9Lou+hO30d3+j66DPfvQoeYREQkJwWEiIjkpIDocm/
cBQwh+i660/fRnb6PLsP6u1AbhIiI5KQ9CBERyUkBISIiOY34gDCzRWb2kpltMrPb4q4nTmY21ciyeNrP1Zl
ZnZp+Ku6a4mVmBmf3JzP5f3LXEzczGmNnDZvaimW0wswvjrilOZvaZ8O9knZn9s5mVxl3TQBvRAWFmB
cA9wDuANHCNmaXjrSpWbcDn3D0NXAB8YoR/HwCfAjbEXcQQ8XfAE+4+BziobvPI7ilw
dXxVjXwRnRAAOcDm9x9s7u3AA8Ci2OuKTbuvt3dV4WvDxJsAI67F/hIYWZTgHcBP4q7lriZWSVwMcE9X
HD3FnffH29VsSsERoV3wxwNbIu5ngE30gNiMvB61nA9I3iDmM3MpgPnAn+It5JY/S/gvwGZuAsZZAmYAj
cCPw0NuPzKzsriLiou7NwD/A9gKbAcOuPsv461q4I30gJAczKwceAT4KwceAT4tLs3xV1PHMzs3cAud38+7lqGiELgj
cD33f1c4DAwYtvszGwswdGGGUAKKDOzD8Zb1cAb6QHRAEzNGp4SjhuxzKyIBzud/dH464nRm8BrjKzL
QSHHi81s3+Kt6RY1QP17t6xR/kwQWCMVJcBr7p7o7u3Ao8C/yXmmgbcSA+IFcBsM5thZsUEjUzLYq4pN
mZmBMeYN7j7d+KuJ07u/gV3n+Lu0wn+Xzzl7sPuF2JfufsO4HUzOzsc9XaCe8aPVFuBC8xsdPh383aGYa
N9YdwFxMnd28zsZmA5QS+E+9y9Luay4vQW4EPAWjNbHY77ors/HmNNMnR8Erg//DG1GfhwzPXExt3
/YGYPA6sIev/9iWF42Q1dakNERHIa6YeYRESkFwoIERHJSQEhIiI5KSBERCQnBYSIiOSkgBA5CTNrN7N7PVW
Y8BO4PYzKab2bqBWp/IQBrR50GI9NFRd18QdxEig017ECKnyMy2mNm3zGytmf3RzM4Mx083s6fM7A
Uz+7WZTQvHTzKzfzWzNeGj49Gj49IMBWb2f8N7C/zSzEaF898S3pvjBTN7MKaPKSOYAkLk5Eb1OMS0JGvaA
XefD3yP4OqvAP8b+Im7vwG4H/j7cPzfA79193EAed68F9gPvC8ffBpwbrufjUX04kd7oTG
qRkzCzQ+5enmP8FuBSd98cXuRwh7uPN7PdQI27t4bjt7v7BDNrBKa4e3PWOqYDT7r77HD480CRu99tZ
k8Ah4CfAz9390MRf1SRbrQHIXJ6vJfX/dGc9bqdrrbBdxHc8fCNwIrwxjQig0YBIXJ6lmQ9Q9/z58/Z903X7yO
uB34etfAzdB572uK3tbqZklgKnu/jTweaASOG4vRiRK+kUicnKjsq5uC8F9mTu6uo41sxcI9gKuCcd9kuDOa
7cS3IWt46qnnwLuNbOPEOwp3ERwN7JcCoB/CkPEgL/XLT5lsKkNQuQhW0QC919d9y1iERBh5hERCQ
n7UGIiEhO2oMQEZGcFBAiIpKTAkJERHJSQIiISE4KCBERyen/A37+3maE333pAAAAAElFTkSuQmCC\n"
        },
        "metadata": {
          "needs_background": "light"
        }
      }
    ]
  }
 ]
}
```