```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "Dataset Importing"
      ],
      "metadata": {
        "id": "FN6Nrg-e2Ao3"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "C5ZEoVkGxE8s",
        "outputId": "32c16dd6-ef5d-41cc-c708-979ac5e1161d"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "dataset = pd.read_csv('/content/drive/MyDrive/spam.csv',
encoding='latin-1')\n",
        "print(dataset.head())\n",
        "print(dataset.info())"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "bdV2vX69xM9a",
        "outputId": "3b4420e9-a9ae-4cf1-cb1c-e4b2635479eb"
      },
      "execution_count": 2,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "     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",
            "  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",
            "<class 'pandas.core.frame.DataFrame'>\n",
            "RangeIndex: 5572 entries, 0 to 5571\n",
            "Data columns (total 5 columns):\n",
            " #   Column      Non-Null Count  Dtype \n",
            "---  ------      --------------  ----- \n",
            " 0   v1          5572 non-null   object\n",
            " 1   v2          5572 non-null   object\n",
            " 2   Unnamed: 2  50 non-null     object\n",
            " 3   Unnamed: 3  12 non-null     object\n",
            " 4   Unnamed: 4  6 non-null      object\n",
            "dtypes: object(5)\n",
            "memory usage: 217.8+ KB\n",
            "None\n"
          ]
        }
      ]
    },
{
  "cell_type": "markdown",
  "source": [],
  "metadata": {
    "id": "Vx9gOSZJ2LDm"
  }
},
{
  "cell_type": "markdown",
  "source": [
    "Importing libraries ,Reading dataset and doing pre-processing"
  ],
  "metadata": {
    "id": "orBHEkB-2Ssw"
  }
},
{
  "cell_type": "code",
  "source": [
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns"
  ],
  "metadata": {
    "id": "OM9xaLiDxyUX"
  },
  "execution_count": 3,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "sns.countplot(data=dataset, x=dataset['v1'])\n"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 297
    },
    "id": "nLsFeFOwx2_F",
    "outputId": "94c589e9-cdb6-443c-ac77-e28c4498fc4b"
  },
  "execution_count": 4,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "<matplotlib.axes._subplots.AxesSubplot at 0x7fd465cbbe50>"
        ]
      },
      "metadata": {},
      "execution_count": 4
    },
    {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ],
            "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEHCAYAAABfkmooAAAABHNCSVQICAgIfAhkiAAAAAlw
SFlzAAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yL
jIsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAARFElEQVR4nO3de7BdZXnH8e+PBLwVJU
pETGjDaGY6IPXSU8DaTi2MgFqFsYg4KtEyjdPBjnZaFTutKMKMVlvE6wwWJGgr4K1Ea8UUsNU
ZBRJRw6WUFKEk5RJNQK2VGnj6x36jm3AO74FmnXOS8/3M7NlrPetdaz97Zs/5nbX2WmunqpAk
6aHsMdsNSJLmPsNCktRlWEiSugwLSVKXKYSFJ6jIsJEldC4fceJJbgB8B9wHbqmoiyROBi4Blw
C3ACVW1NUmAs4EXARNAAeuSrK6qrVO94LJly1i
+2bNCwaH63qr4/Nn8qcFlVvTvJqW3+rcALJWrcRjwUeCwFi6nARNAAeuSrK6qrVO94LJly1i
7du0w70aSdlNJbp1q2WwchjoW2WwchjoW2L5nsAo4bqx+QY18E9gnyf7A0cCaqtrSAmINcEyb
YVHAV5KsS7Ky1farqtvb9B3Afm16EXBb2LobW22q+gMkWZlkbZK1mzZt6pvZDzLkbO1Wfs
W1K8mRgTZJ/G19YVZVkp9xvpKrOAc4BmJiY8B4mkrQTDbpnUVWb2vNdw0eBQ4E72+ebf
gm4ICx1Ze22lR1SdIMGSwskjwuyd7bp4GjgGuB1cCKNmwFcEnXwXcF0CaLTFXFKe
yqG3n0qH6liQ92JCHofYDPj86I5aFwN9X1ZeTXA1cnORk4FbghDb+S4xOm93A6NTZ1wFU2ZYk
7wKubuNOr6otA/YtSdpBdsdblE9MTLCu3dQHJIDVDNTLbMK7glSV2GhSSpayau4N4l/fqbL
5jtFjQHrXvvSbPdgjQr3LOQJHUZFpkKLsNCktRlWEiSugwLSVKXKYSFJ6jIsJEldhoUkqcuwkC
R1GRaSpC7DQpLUZVhIkroMC01lS2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAt
JUpdhIUnqMiwkSV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXKYSFJ6jIsJEldhoUkqcuwkC
R1GRaSpC7DQpLUZVhIkroM
C0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIUnqMiwkSV2GhSSpy
7CQJHUZFpKkLsNCktRlWEiSugwLSVKXKYSFJ6jIsJEldhoUkqcuwkCR1GRaSpC7DQpLUZVhIkr
oMC0lSl2EhSeoaLCySPDrJVUm+k+S6JO9s9QOTXJlkQ5KLkuzV6o9q8xva8mVjr3Vqq9+Y5OilkuzuzV6o9q8xva8mVj23pbq9+Y5Oi
hepYkTW7IPYt7gSOq6pnAs4BjkhwOvAc4q6qeBmwFTm7jTwa2tvpZbRxJDgJOAA4GjgE+kmTB
gH1LknYwWFjUyI/b7J7tUcARwGda/QLgK2360Jz/TD4vSKtzF0mJ5G4t0YgHx7q57tqGG3CqcS3q
...
bojuA/dr0EuC2sdU2ttpU9R1fY2WStUnWbt68eaf2L0nz3eBhkeSXgM8Cb6qqH44vq6oCame8
TlWdU1UTVTWxeLE/tSFJO9OgYZFkT0ZB8XdV9blWvrMdXqI939Xqm4ADxlZf2mpT1SVJM2TIs
6ECnAvcUFV/M7ZoNbD9jKYVwCVj9ZPaWVGHA/e0w1WXAkclWdS+2D6q1SRJM2TIO8c+D3gNsD
7Jt1vtz4F3AxcnORm4ldGPKgF8CXgRsAH4CfA6GN0OPcm7gKvbuNPbLdIlSTNksLCoqq8DmWL
xkZOML+CUKbZ1HnDezutOkvRweAW3JKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIUnqMiwk
SV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUkqcuwkCR1GRaSpC7DQ
pLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIUnqMi
wkSV2GhSSpy7CQJHUNFhZJzktyV5Jrx2pPTLImyU3teVGrJ8kHkmxI8t0kzxlbZ0Ubf1OSFUP
1K0ma2pB7FucDx+xQOxW4rKqWA5e1eYAXAsvbYyXwURiFC3AacBhwKHDa9oCRJM2cwcKiqv4V
2LJD+VhgVZteBRw4J7JNkf+BoYE1VbamqrcAaHhxAkqSBzfR3tV1e1t+g5gvza9B
LhtbNzGVpuq/iBJViZZm2Tt5s2bd27XkjTPzdoX3FVVQO3E7Z1TVRNVbF48eKdtVlJEjMfFn
e2w0u057tafRNwwNi4pa02VV2SNINmOixWA9vPaFoBXDJWP6mdFXU4cE87XHUpcFSSRe2L7aN
aTZI0gxYOteEknwKeD+ybZCOjs5reDVyc5GTgVuCENvm7jTq+q
Hb80lyQNbLCwqKpXTrHoyEnGFnDKFNs5DzhvJ7YmSXqYvIJbktRlWEiSugwLSVKXYSFJ6jIsJ
Eldg50NJWkY/3n6IbPdguagX377+kG3756FJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIU
nqMiwkSV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUkqcuwkCR1GRa
SpC7DQpLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1LXLhEWSY5LcmGRD
klNnux9Jmk92ibBIsgD4MPBC4CDglUkOmt2uJGn+2CXCAjgU2FBVN1fV/cBFwLGz3JMkzRu7Slgs
O0GpmkJcNvY/EbgsPEBSVYCK9vsj5PcOEO9zQf7At+f7SbmgrxvxWy3oAfys7ndadkZW/mVqR
bsKmHRVVXnAOfMdh+7oyRrq2pitvuQduRnc+bsKoehNgEHjM0vbTVJ0gzYVcLiamB5kgOT7AW
cCKye5Z4kad7YJQ5DVdW2JG8ALgUgUWAOdV1XWz3NZ84uE9zVV+NmdIqmq2e5AkzXG7ymEoSdIs
MiwkSV2GxTyWZFmSa2e7D0lzn2EhSeoyLLQgyblJrkvypSSPTfLlSc5LcmGRD7xnvdnbOE7qzwj
STfTHJzkucnOS/JDUnOn+X3oV1ckscl+cf2ySu3JLkr5KsT3JVkqe3sS9Jcm
L9Wv0dSVYl+VqSW5O8bGz9LyfZc3bf5a7LsNBNy4MNVdTBwN3A8cCZ3ADIB/D7R5PYn+/pP
F/oTRtS5nAQcDhyR51ox2rt3NMcB/VdUzq+oZwb/Z6qOgT4EPD+Vvs6cHhVPZvReVAwVhxc9Pv
pwFHAC8FPglc0db/H+DFw7+N3ZNhoe9U9bfb9DpgGGfCM9p/ZeuBVt9cCdVbW+q
u4HrmvrSo/UeuAFSd6T5Ler6p5W/9TY83gPdlg0vYZfTMP/Iz+U1X9rG1vAb8InfX4GX3EDA
vdOzZ9H6MLNc8H3tD+G3sn8OhJxt+/w7r3s4tc5Km5qar+HXgOoz/qZyR5e2uanz/ZF48Pa8weBD7X
P6OuZ5DPa/on5Wf3iYjI/o/8PhoUmszdwezu++6rZbkbzQ5KnAj+pqk8BdqR8ShHKj+pqk8C72UUHACvGHv+Rpt+
Ar+4P5y3Ap4Bpqwm85fAlcDm9rz37LajeeIQ4L1J7gd+Bvwj8BxdoU1ZLvMtpjeGUb+w7g00m2A
pcDB858u/OLt/uQNGcluQWYqCp/s2KWeRhKktTlnoUkqcs9C0lSl2EhSeoyLCRJXYaFNMPaaPGLS
ruTvLF2e5Fmi7DQpp57wVeM9t9NJHZcBQnQMLL
Y/AmtJu1yvN2HNJCquibJk9s9jxeegnsV2oUZFtKwLgI+BuwL
/M4s9yI9Yn5nIQ2oqq5jdL+PTVV122z3Iz1sHusmxCV/aSr5jZtTYT4vQtE69EX/UHENL1J9Ylb5sJvaKmP7SDF
VhIkroMC0lSl2EhSeoyLCRJXYaFJKnr/wAj4WzKPZKTWQAAAABJRU5ErkJggg==\n"
        },
        "metadata": {
          "needs_background": "light"
        }
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "text = dataset.loc[:, 'v2']\n",
      "classification = dataset.loc[:, 'v1']\n",
      "print(text)\n",
      "print(classification)"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
        "id": "7BBr99i7x4ON",
        "outputId": "a2448d6f-ed4b-4de9-a6fc-f635e8ba422d"
      },
      "execution_count": 5,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "0       Go until jurong point, crazy.. Available only
...\n",
            "1                           Ok lar... Joking wif u
oni...\n",
            "2       Free entry in 2 a wkly comp to win FA Cup
fina...\n",
            "3       U dun say so early hor... U c already then
say...\n",
            "4       Nah I don't think he goes to usf, he lives
aro...\n",
            "                              ...
\n",
            "5567    This is the 2nd time we have tried 2 contact
u...\n",
            "5568                Will Ì_ b going to esplanade fr
home?\n",
            "5569    Pity, * was in mood for that. So...any other
s...\n",
            "5570    The guy did some bitching but I acted like
i'd...\n",
            "5571                           Rofl. Its true to its
name\n",
            "Name: v2, Length: 5572, dtype: object\n",
            "0        ham\n",
            "1        ham\n",
            "2       spam\n",
            "3        ham\n",
            "4        ham\n",
            "        ... \n",
            "5567    spam\n",
            "5568     ham\n",
            "5569     ham\n",
            "5570     ham\n",
            "5571     ham\n",
            "Name: v1, Length: 5572, dtype: object\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from nltk import word_tokenize\n",
        "from sklearn.model_selection import train_test_split\n",
        "import nltk\n",
        "nltk.download('punkt')\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "nsIN45Usx9bo",
        "outputId": "93a2fecf-dace-40ca-ff1f-a5964707e845"
      },
      "execution_count": 10,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "[nltk_data] Downloading package punkt to
/root/nltk_data...\n",
            "[nltk_data]   Unzipping tokenizers/punkt.zip.\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "True"
            ]
          },
          "metadata": {},
          "execution_count": 10
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "x_train, x_test, y_train, y_test = train_test_split(text,
classification, test_size=0.2, random_state=42)"
      ],
      "metadata": {
        "id": "u4Wg4_KmyDYq"
      },
      "execution_count": 11,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "text_length = []\n",
        "for i in x_train :\n",
        "  text_length.append(len(word_tokenize(i)))"
      ],
      "metadata": {
        "id": "0fXLb52iyEan"
      },
      "execution_count": 12,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "print(max(text_length))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "a6Hp6SLiyHs5",
        "outputId": "3476da61-3c55-4569-f650-1f736c2b5d9a"
      },
      "execution_count": 13,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "220\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from keras.preprocessing.text import Tokenizer\n"
      ],
      "metadata": {
        "id": "w43PINNMylli"
      },
      "execution_count": 14,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "max_sequence_length = 38\n",
        "\n",
        "tok = Tokenizer()\n",
        "tok.fit_on_texts(x_train.values)"
      ],
      "metadata": {
        "id": "UsB84_wvyu3X"
      },
      "execution_count": 15,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "vocab_length = len(tok.word_index)"
      ],
      "metadata": {
        "id": "dSSZ1hYgywz3"
      },
      "execution_count": 16,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "vocab_length = len(tok.word_index)"
      ],
      "metadata": {
        "id": "KuBR5BKAy1v2"
      },
      "execution_count": 17,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "x_train_sequences = tok.texts_to_sequences(x_train.values)\n",
        "x_test_sequences = tok.texts_to_sequences(x_test.values)"
      ],
      "metadata": {
        "id": "eWFlz3qwy3YP"
      },
      "execution_count": 18,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.utils import pad_sequences\n"
      ],
      "metadata": {
        "id": "5Zxb5WjFy8EY"
      },
      "execution_count": 19,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "x_train = pad_sequences(x_train_sequences,
maxlen=max_sequence_length)\n",
        "x_test = pad_sequences(x_test_sequences,
maxlen=max_sequence_length)"
      ],
      "metadata": {
        "id": "FWZOKsdey-97"
      },
      "execution_count": 22,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "x_train[:2]"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "1v5wdMbXzCzZ",
        "outputId": "98bc89d5-833a-46c4-805a-a551f2d34943"
      },
      "execution_count": 21,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "array([[   0,    0,    0,    0,    0,    0,    0,    0,
0,    0,    0,\n",
              "           0,    0,    0,    0,    0,    0,    0,    0,
38,   30,    8,\n",
              "           5,  273, 1989,   81,  116,   26,   11, 1656,
322,   10,   53,\n",
              "          18,  299,   30,  349, 1990],\n",
              "       [   0,    0,    0,    0,    0,    0,    0,    0,
0,    0,    0,\n",
              "           0,    0,    0,    0,  799,   15, 2555, 1442,
1127,  192, 2556,\n",
              "         171,   12,   98, 1991,   44,  195, 1657, 2557,
1992, 2558,   21,\n",
              "           9,    4,  203, 1025,  225]], dtype=int32)"
            ]
          },
          "metadata": {},
          "execution_count": 21
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "y_train.values"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "oDJh_fekzDYt",
        "outputId": "207fc2ca-a243-4b93-e653-a986d85fe430"
      },
      "execution_count": 23,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "array(['ham', 'spam', 'ham', ..., 'ham', 'ham', 'ham'],
dtype=object)"
            ]
          },
          "metadata": {},
          "execution_count": 23
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from sklearn.preprocessing import LabelEncoder\n"
      ],
      "metadata": {
        "id": "sJd1reTPzDf-"
      },
      "execution_count": 24,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "le = LabelEncoder()\n",
        "y_train = le.fit_transform(y_train)\n",
        "y_test = le.fit_transform(y_test)\n",
        "print(y_train)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Ph1Vckd2zDqb",
        "outputId": "603d8c4e-3ee8-46a7-a926-a6b22a491ea6"
      },
      "execution_count": 25,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "[0 1 0 ... 0 0 0]\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from keras.models import Model, load_model\n",
        "from keras.layers import LSTM, Activation, Dense, Dropout,
Input, Embedding\n",
        "from keras.optimizers import RMSprop"
      ],
      "metadata": {
        "id": "vwJIJgsUzYnL"
      },
      "execution_count": 26,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "Creating models and Adding layers"
      ],
      "metadata": {
        "id": "rKYqkuIs2e0b"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "def create_model(vocab_len, max_seq_len):\n",
        "    inputs = Input(name='inputs', shape=[max_seq_len])   #None,
150\n",
        "    layer = Embedding(vocab_length + 1, 50,
input_length=max_seq_len)(inputs) #None, 150, 50\n",
        "    layer = LSTM(64)(layer)  #None, 64\n",
        "    layer = Dense(256,name='FC1')(layer) #None, 256\n",
        "    layer = Activation('relu')(layer) #None, 256\n",
        "    layer = Dropout(0.5)(layer) #None, 256\n",
        "    layer = Dense(1,name='out_layer')(layer) #None, 1\n",
        "    layer = Activation('sigmoid')(layer) #None, 1\n",
        "    model = Model(inputs=inputs,outputs=layer)\n",
        "
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),
metrics=['acc'])\n",
        "    return model\n",
        "\n",
        "model = create_model(vocab_length, max_sequence_length)\n",
        "model.summary()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "AmH4R-9ezYsu",
        "outputId": "3196995a-af8a-4004-95c0-19322f8cff03"
      },
      "execution_count": 27,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"model\"\n",

"_________________________________________________________________\n",
            " Layer (type)                Output Shape              Param
#   \n",

"=================================================================\n",
            " inputs (InputLayer)         [(None, 38)]              0
\n",
            "
\n",
            " embedding (Embedding)       (None, 38, 50)
397750    \n",
            "
\n",
            " lstm (LSTM)                 (None, 64)                29440
\n",
            "
\n",
            " FC1 (Dense)                 (None, 256)               16640
\n",
            "
\n",
            " activation (Activation)     (None, 256)               0
\n",
            "
\n",
            " dropout (Dropout)           (None, 256)               0
\n",
            "
\n",
            " out_layer (Dense)           (None, 1)                 257
\n",
            "
\n",
            " activation_1 (Activation)   (None, 1)                 0
\n",
            "
\n",

"=================================================================\n",
            "Total params: 444,087\n",
            "Trainable params: 444,087\n",
            "Non-trainable params: 0\n",

"_________________________________________________________________\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Compiling model"
      ],
      "metadata": {
        "id": "YjYw2Q3F2wda"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from keras.callbacks import EarlyStopping, ModelCheckpoint,
TensorBoard"
      ],
      "metadata": {
        "id": "aRzZ6AnKzfXl"
      },
      "execution_count": 28,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "history = model.fit(x_train, y_train, batch_size=128, epochs=20,
validation_split=0.2)\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "_qnSyCuYzrdx",
        "outputId": "66f4b8db-5ff3-443f-c6a4-1109affa5d97"
      },
      "execution_count": 29,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Epoch 1/20\n",
            "28/28 [==============================] - 5s 98ms/step -
loss: 0.2984 - acc: 0.8741 - val_loss: 0.1544 - val_acc: 0.9552\n",
            "Epoch 2/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 0.0803 - acc: 0.9820 - val_loss: 0.0573 - val_acc: 0.9821\n",
            "Epoch 3/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 0.0268 - acc: 0.9924 - val_loss: 0.0419 - val_acc: 0.9865\n",
            "Epoch 4/20\n",
            "28/28 [==============================] - 3s 98ms/step -
loss: 0.0151 - acc: 0.9961 - val_loss: 0.0412 - val_acc: 0.9843\n",
            "Epoch 5/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 0.0083 - acc: 0.9969 - val_loss: 0.0678 - val_acc: 0.9843\n",
            "Epoch 6/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 0.0052 - acc: 0.9983 - val_loss: 0.0690 - val_acc: 0.9854\n",
            "Epoch 7/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 4.3604e-04 - acc: 1.0000 - val_loss: 0.0707 - val_acc: 0.9865\n",
            "Epoch 8/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 4.3695e-05 - acc: 1.0000 - val_loss: 0.0848 - val_acc: 0.9888\n",
            "Epoch 9/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 0.0029 - acc: 0.9994 - val_loss: 0.0913 - val_acc: 0.9798\n",
            "Epoch 10/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 2.9656e-04 - acc: 1.0000 - val_loss: 0.0992 - val_acc: 0.9832\n",
            "Epoch 11/20\n",
            "28/28 [==============================] - 2s 76ms/step -
loss: 1.8744e-05 - acc: 1.0000 - val_loss: 0.1156 - val_acc: 0.9854\n",
            "Epoch 12/20\n",
            "28/28 [==============================] - 2s 76ms/step -
loss: 2.5507e-06 - acc: 1.0000 - val_loss: 0.1101 - val_acc: 0.9888\n",
            "Epoch 13/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 1.1500e-06 - acc: 1.0000 - val_loss: 0.1304 - val_acc: 0.9865\n",
            "Epoch 14/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 3.2071e-07 - acc: 1.0000 - val_loss: 0.2487 - val_acc: 0.9821\n",
            "Epoch 15/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 0.0046 - acc: 0.9994 - val_loss: 0.0985 - val_acc: 0.9832\n",
            "Epoch 16/20\n",
            "28/28 [==============================] - 2s 76ms/step -
loss: 1.4251e-04 - acc: 1.0000 - val_loss: 0.1183 - val_acc: 0.9854\n",
            "Epoch 17/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 8.2607e-06 - acc: 1.0000 - val_loss: 0.1241 - val_acc: 0.9854\n",
            "Epoch 18/20\n",
            "28/28 [==============================] - 2s 74ms/step -
loss: 3.0136e-06 - acc: 1.0000 - val_loss: 0.1288 - val_acc: 0.9843\n",
            "Epoch 19/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 1.9952e-06 - acc: 1.0000 - val_loss: 0.1374 - val_acc: 0.9843\n",
            "Epoch 20/20\n",
            "28/28 [==============================] - 2s 75ms/step -
loss: 4.4795e-07 - acc: 1.0000 - val_loss: 0.1438 - val_acc: 0.9854\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Fitting and Saving the model"
      ],
      "metadata": {
        "id": "inuX6SVu22RM"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "history_dict = history.history\n",
        "\n",
        "# list all data in history\n",
        "print(history_dict.keys())\n",
        "\n",
        "# summarize history for loss\n",
        "plt.plot(history_dict['loss'])\n",
        "plt.plot(history_dict['val_loss'])\n",
        "plt.title('Training and Validation Loss')\n",
        "plt.ylabel('loss')\n",
        "plt.xlabel('epoch')\n",
        "plt.legend(['train', 'test'], loc='upper left')\n",
        "plt.show()\n",
        "\n",
        "# summarize history for accuracy\n",
        "plt.plot(history_dict['acc'])\n",
        "plt.plot(history_dict['val_acc'])\n",
        "plt.title('Training and Validation Accuracy')\n",
        "plt.ylabel('accuracy')\n",
        "plt.xlabel('epoch')\n",
        "plt.legend(['train', 'test'], loc='upper left')\n",
        "plt.show()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 590
        },
        "id": "mQWNKEdE0Gne",
        "outputId": "f549e9ff-5568-4eb3-c33f-92be2443efaa"
      },
      "execution_count": 30,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])\n"
          ]
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ],
            "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYgAAAEWCAYAAAB8LwAVAAAABHNCSVQICAgIfAhkiAAAAAlw
```

SFlzAAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yL
jIsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAgAElEQVR4nO3deXxcdbn48c8zSZpJmq
1NQpN0S1ra0JZCgRYEWkBZLMgqCGUTEcEFrvpTucJLRMXrlavX5SqLIKAguyxatbIUKDu0pS1
LN7rQJVubLknaZk+e3x/fM+k0nUkmy8kk+f9es1rzpxlzpOTZJ4531VUFWOMMaYzX7wDMMYY
MzBZgjDGGBOSJQhjjDEhWYIwxhUYIwxhgTkiUIY4wxIVmCMMYYE5IlCBN1IvJvEbmqv/eNJxHZJCKnR
eF9F4nIV7zly0XjV4xXkhUj27cV5xonIXhFJ6m2sJvFZgjAheR8egUe7iDQEvb9ve7iDQEvb68J++lqmeq6oP9ve
9AJCI3ichrIdbniUiziBwe6XUp6iOqekY/Xup6iOqekY/xXVAQlPVLaqaoapt/fH+nc6lInJof7+viT1LECY
k78MjQ1UzgC3AOUHrHgnsJyLJ8YtyQHoYOEFESjqsvwRYqajOmLwEPdBeq7stOmLwEPdxdEp5i+JyBtBr08XkT
UiUisidwASG2iiLzsxbdDRB4RkRxv21+AcK3T5GIzBeRXSKyXkDrXSK3T5GIzBeRXSKyXkSuDXr
vH4vIkyLHdtVorIzHDXIBwRyfbeo9q7lreIiM/bdqiIvOr9RkZBFMUwRyfbeo9q7lreIiM/bdqiIvOr9bDtE5AlvvvYjIb0Rku4jUiciH
PbkLM31jCcL0RgEwEhgPXIf7O/qT93ocL0RgEwEhgPXIf7O/qT93oc0ADc0cXxxwFrgTzgF8D9IiK92PdRYDGQC/yYg2+gwxe4AOyBNU
0US42XA1cAhwDDgewAiMhW23v/Iu98IT/UPQ8GxyIipcAML96eX0qvAe+QBzwC/QBzwC34K7FBuDE4F
2An3vxTQHG4q4JqnolB94F/iLEKR4HyrzjLwL+W0Q+E7T93l/QEGH8HsgG5gAnIxLmld
7234KvACMwF3b33vrzwBOAiZ7x14M7OzFuU1vqKo97NHlA9gEnOYtnwI0A+4uHndgsZtUdmbvhNhUVKQc8
irf8JWB90LZ0QIGCnuyL+3BtBdKDKtj8MPBzhzxQqxluCXn8DeM5bvhV4PGjbcGAOvbDkjAE9AnuaLG6KnN8bt
uAE7/XPgL/38lq94S1/EXgnaD/BfaB/Jcz7ngQ9e9cyGZdDUeM5oDMoO0/U79F4Xe7
i0bSrQ0MW1VeDQTuuSvGs2NWjdV4FF3vJDwL3AmE7HfQb4GPgU4Iv3/8JQe9gdhOmNalVtDLw
QkXQRuccrrNqgDXgNyJHwLmarAgqrWe4sZPdy3CNgVtA5ga7iAI4yxKmi5PiimouD3VtV9dPEt
1ovpr8AXvbudy3EfgL25VW9cfxt1pRCJwLfcErdsMjA563fna+KVn9
U95QIr3vqHO8Z+4pLfYK8L6MoCqvoy7W7kT2C4i4C4i94pIVg/Oa/rAEoTpXUs
EABJWRR0ElMFJE0oPWje1i/77EWBn83t45c7s55kFcccjpQCbwjz7G0TkG4cCf979xv5fp3vt
e0ek9uxq2uQJ3LTOD1o0DyruJqSd2AC24orWDzqGqVap6raowW4e4s7hKvJZSq/k5Vj8HduwG
buzHuEwXLEGY/pCJK9JK0uvEZGRwEJnCjF+BRwtojMFpEpFhwG10/7zO
lCDKzZ5XFWb+xjHv4BpIvJ575nrVbv7N3FFbGc/CH6Dzc2f9BVHUr8Bbwcx Hi8gRwD
W4u5DeGua9lO19E/N66J4GfiUimN4GfiUimN4GfiUimxhM4h4h8IaiyfjcuobWLyCwROU5EqBAv8QCPQ3oe4TA9
YgjD94bdAGu5b4jvAczE67+XA8bjinv8CngAwuzb6xhVVwPa6SuRL3AVbWzTGKK1Ya7z33
KQ5V3QF8Abgd9/NOAt4M2M2uZnwNFALS6ZPNPpLX4O3CIiNSLyvRCR4ZiLvRCnuBRXXL1EBPAv8SFUX9
CtxiTDwuBr4D9yH/EbgDdz1fMDbxwrojsxVWCf0tVNwKCwMB9x13mem
Oe1zRyjapG/Q7GmKHA7iDMoOUVP0wUEZ+IzvXlEyxX
DMiZxWBGTMcaYkKyIyRhjTEgJU8SUl5enEcnF6xR9a8r6Ht7DGGYjXCaK4uJilS5fG
OwxjjBlURGzuG1WxGSMMSYkSxDGGGNCsgRhjDEmpISpgzEmcbS0tFBWVkZjY2O8Q4kZfr+f0aNHk5KSEvE4lw
4aUlJR4h2h2KMMSPRBBRTRTAe79b/w0QYau05kPP7/+/w/pe
uQvSfH3quqM1V1Zn5+yI6OEd5aEDl1W18i+pugkiJqaGu66Gu66664eH3fWWWdRU1PTP/3++zyY
RsMDGPPPB6+YPJyP2q+rr+PNEC8MliMNbWrhPSggULyMyMxhgTiWgmCxqNxPoDgU8
4JeIYup7C8HHcROu9ObbXkkDSFp1BC8MliNbWrhPSggULyMyMxhgTiWgmCxqNxPoDgU8
8aZpnIebKaqDiEwKevk5evk5YJ23PB+G2MGPGD
GbNmsWcOXT4347jjDPOxeyMslCtdeey3Tpk3jjDPOoNmMsKg1c1XVhG5AXge18z1AVVdKSK3AUtVdT7wLVxP5qS
3jjDPOoKGhISqxGmNMsKg1c1XVhG5AXge18z1AVVdKSK3AUtVdT7wg4ichpvMfDdwlXfsS
5ElgFtALXq2pbX+L2AfJ3xzUc2WLKaxV0ZsXpWLKaxV0ZsXp12FEhL6VlDqalFWzzonGld7nP77bfz0U
WsTnPvc5Pvro47mqA888AAjR46KjR4nbm3vAe6xbt4447jjHjH8
fTTXHHFFT2K1Rhjeqiq/SBUdQGwoNO6W4KWHwSODHPuz4CfdKN8
53PPvsswBs3bqzvZ3pt+VW0j2/c0cvj
obHxRbi46fPjwuVFixaxcOHCTJkymGz0n+jiP+mwXhYqA/Fgmry8xUqZ4m/uYwQYH4dWzx
kkJrFCqqMzMz2bNnT8httbW1jBgxgvT0dNasWcM777zTz3NkjOl/ZYbg4c5MdDNnWgJ7ywBcNj5+m
Zszc3NzOfHEE2r3cRjmBgdcc9d3T+HTyy1/Yd5YhKj/XmCtAM+tSn+vXcxhjTFwS/XMmL
0184RBq1evZsqUdHl29QQy0c9GHQCjHqIv15jTEmQETeU9WZWz5e0/3GUofE1ROOOkqSKmKL
VF8IYYwYjSxEd5aLTl8IY4wZjCxEBP3OcsYYMxhZgvCkJEVuA1jjBmZLEF47LKH6C8BAR3luwPliCMM
k5LkIo7WtnQ1WWgQo/d+SqbfDfQP89re/hu+iCMcaYBqG9n/DQhnjKCMF8HiBbsjnd/PGoRHQqyM
YOV9aT2pCRHEMDD/9+9ynhh/Dkk0/S8hpumr/S1NTE+eefz6+//evecc/Vu3TTePwX4lZ4fCxeoegqTF3GwIeR0M9Db0OWOk03NmbH1twmSuA+p/GOYGQyqeNetx2vL7i1u9v2Hxb3sinwzPQOyBZtKlIlMGiNzmp1pwhyTbLF1lZkfEQqGnqmpMSYxaJYPo2LgvyfvwBD8O4Hz2
S2IMuCU3ZRsZstSN0ltbSTe1kkakUOL9DcJz6iPj3DLYpSZcEWl9V2YYH9gvTNPsXCXa2vYZl+4Tz/5Wj8MAZ/q
lpb6U+f+G/bx1u9pyEV5OLrPNEDo3htptPOnJSaRjOoCXwBWWNLwvfGHhWixZlPFOJPnsdFJWGYvdY1m0ZZVsYb2Wp2TzekkjazeAE1RLQFujJc
5RT5ZoPpybm7Yfn1PD+bNJ8vrO8vGuV2BLwhMqw8Y7PqDL6fWjkxr2PLXjQF6L5qFozBDA09mpJhmM33XeSJY4b6E9jXM2hdmPR2l7hpnbVDTj
8YHqfRBKrzbU3EW3dzL56whnHq53CKC3QHMIX8RTAC1K9qz4J8a6gLaH3jf1Nf9YOkTAjX1HiU1q6rVsVjGNQtdX2AQjiCvrqiLyMuegM
Y4Ej13bzFbXCcTZQfXZYuRGu6oVYbyDs8vnkKBDdkuIjdOWfcN2GDdQVePxoQLhEoRbM9ySOgpkpGkL3XqmhAtuS6jh

9k27ZtVFRU8OlPf5q8vDxeeeWVfo3LGGO6M3QSxL9vgqoPw272o0xoanNDbSRFeGNVMB3OvL3
LXYKH+37hhRd46qmnWLx4MarKueeey2uvvUZ1dTVFRUX861//AtwYTdnZ2fz617/mlVdeIS8v
L+If0xhj+osVMQURIaqV1C+88AIvvPACRx11FEcffTRr1qxh3bp1TJ8+nRdffJHvf//7vP766
2RnZ0ctBmOMidTQuYPo5pu+AGVVe/Cn+BifO7zLfXtLVbn55pv56le/etC2ZcuWsWDBAm655
ZOPfVUbr311hDvYIwxsWN3EEGi0VkueLjvz372zzwwAPs3bsXgPLycrZv305FRQWp6elcffXV
V3HjjjSxbtuygY40xJtaGzh1EBFKSfOxtau3X9wwe7vvcc8/lyiuv5OSjjx5J04KB92wCsvLCp
13PjjTfi8/lISUnh7rvvBuC6665j7ty5FBUVWSW1MSbmbLjvIFW1tVYDnQUTa4nNHaPa
1M23HeEotVZzhhjBiNLEEGi1VnOGGMGo4RPED0pQhvMCSJRigqNMQNHQicIv9/Pzp07I/7wHK
wzy6kqO3fuxO/3xzsUY0wCSehWTGPGjKGsrIzq6uqIj6muaWDf9mR2pKVEMbL+5/f7GTNmTLz
DMMYkkKgmCBGZC/wfkATcp6q3d9r+HeArQCtDXxZVTd729qAwNgYW1T13J6ePyUlhZKSkh4d
c/3/LmJKYRZ3Xn5ET09njDEJJWoJQkSSgDuB04EyYImIzFfVVUG7LQdmqmq9iHwd+AVwibetQQ
VVnRCu+cApz/FTUNsT6tMYYM+BEsw7iWGC9qm5U1WbgceC84B1U9SWqqvDYn
ob4x2GMcbEXTQTxGhga9DrMm9dRdcNv5/w567ReRpSLyjoicH+oA/J4OZl42dpT+oZulKU42dbXSO
tg7AlkzFxservsPnteEdhomBAtGISkSuAmcAvg1aP93Ve9V1ZmqOmdj5OVe9V1ZmqOjM/P79f
YinI9tOuUL23qV/ez5iEpgr/+DYs+nm8IzFREM0EUQ6MDXo9xlt3t3ABE5DfgBcK6qdnwqq2q59
7wRWAQcFcVYOxRmu9Fjd+xp7OdZlE/2jaWAwy7Y8XG8IzFREM0EsQSYJCITRCQVuBSYH7yDiBwF
Elh+1B60eISKq3nAecCARXbkdNYXYaAJU1liCM6VaFG3mYPZXQWBvfWEy/i1qCUNVW4AbgWWA
18KSqrhSR20Qk0GT1AG8FcRWSEigQQyBVgqIu8DrwC3d2r9FDX77yCsJZMx3apYvn95x7r4
xWGiIqr9IFR1AbCg07pbg5Zl7ZPC3PcW8D0aMYWTnZaCmkpSdaSyZhIVCyH9Fyo3wnVa2FMyEFBz
SA1ICqpBxIRoTDbb3UQxnSnvR0qVsBhZ0PSMNixNt4RmX5mCSKEgmy/FTEZ05ldG6GpDsbMgt
xD3R2ESSiWIEIozE6zOwhjuhOofxh9NORNtgSRgCxBhFCY7Wf7nibrLGdMVyqWQXIa5JVCfin
UbIYW+2KVSCxBhFCQ7aetXdmxtzneoRgzcFUsh8IjICnZJQhth53r4x2V6UeWIEIoynFNXW3Q
PmPCaG+Dyveh6Gj3Oq/UPVtFdUKxBBFCQZbrLGdNXY0Jo3ottNRDkTfAQe6hID6rh0gwliBC6
LiDqLE7CGNCClRQBxJEih9yxluCSDCWIELITkvBn+KzOwhjwqlYDsMy3Z1DQH6pjcmUYCxBhO
A6y6VRWWcJwpiQKpZB0QzwBX2E5Je6Suq21vjFZfqVJYgwCrP9VFoRkzEHa22Gqo9cggiWVwp
tza65q0kIliDCKMj2WxGTMaFUr4a2pv31DwH5Xkum6jWxj8lEhSWIMAqz/Wzb00Rbu8Y7FGMG
lnJviO9AE9eAvEnu2SqqE4YliDAKs9Noa1eq99jMcsYcoGI5+HNgRPGB6/3ZkFloFdUJxBJEG
DYvhDFhVCx3xUsiB2/LL7U7iARiCSKMjpnlrlrB7CmP1aGmH7KjdAXyh5XlNXtaLZRGAJIgybm9
qYELZ9BO2tB1dQB+RPhua9UHfQ9PNmELIEEUZOegqpyT6qrIjJmP0696DuLDAmkxUzJQRLEGG
ICEU5aVTYHYQx+1Ush+GHQNbo0NsDTV2tojohWILoQkGW9YUw5gDly8JXUAMMz4e0EXYHkSAs
QXShMMcShDEdmva64bzDFS+BSx7MoP3aRA4VowBeRPtt7UC
cISRBcKvM5yO/ZaZzljqPB6UBfO6Hq/vFKo3wn7dkY/JhNVliC6UJRt80IY06Fiuauczhz
V9X
75NrtcoohqghCRuSKyVkTWi8hNIbZ/R0RWicgHIvKSiIwP2naViKzzHldFM85wCrwEYfUQxC
/B3V38q2pa6KIWoIQkSTgTuBMYCpwqYhM7bTbcmCmqh4BPAj4DjgWWOBHIjWrGG
Y72pjfE01Li5HiJJEFljICXdKqoTQDTvII4F1qvqRlVtBh4HzgveQVVfUdV67+U7wBhv+bPAi
6q6S1V3Ay8Cc6MYNeNBPEaGBr0Osyb1041wD/7s
mxInKdiCwVkaXV1dV9DPdgbmY5v91BGNNdD+rO8kqh2u4gBrsBUUktIlcAM4Ff9uQ4Vb1XVWe
q6sz8/PyoxFaYnWZ1EMZULHPDe6ePjGz//LEGtWgmiCXAJBEpEZFhwDxgvAOInUcA8uOWwP2vQ8cIaIj
berJsbFgdxDGEHkFdUD+Ye7Z6iEGtWgmiCXAJBEpEZFhwDxgfvAOInIUcA8uOWwP2vQ8cIaIj
PAqp8/w1sVcQbafbdZZzgxl+3ZCzZaeJYg8G5MpEUQtQahqK3AD7oN9NfCkqq4E5Fxvt1
8CGcBfRWSFiMz3jt0F/BSXZJYAt3nrYq4g22/DbRhTtxN2b+m6/iFY/mT3bBXVg15Ux+eFGOUtGzOkBBJE4ZGRH5OUAiMnWkX1IDcgKqkHssKc
QGc5a+pqhqiK5ZA7CfxZPTsuf7L1ph7kLEF0I9BZrqLGKqrNEFWxrPsB+kLJPwx2fQKtzf0fk
4kJSxDdCHSWq6qzBGGGoLpK2FPZswrqLxS0DbYtaH/4zIxYQmiG9ZZzgxplSvcc28SRP5k92
wV1YOWJYgIFGT7qbQRXc1QVL4MMAcFR/T82NYwS2bN3ZEJeG7dW/Ra49/z
YYemQM9YqqgcxSxARKLTOcmYoUu15D+rO8g+zO4hBzBJEBAqz/bS2Kzuts5wZwZSmq3Qv0OKOpm
Brmu5E2Gneugva3/4jIxYwkiAjYvhBmSAh3ketPENSC/FFoboWZz/8RkYsoSRAQCM8vZvBBmS
KlYDr4UGHV4798jMCaTFTMNSpYgIlCYvX+/4DWOGoqJKf2/j0CTV2tonpQsgQRgZHDh
Es2WfzQpihQxUqVvRsgL5Q0kZAxii7gxikLEFEINBZrsISHBkqdm2Eptq+tWAKyLMxmQYrSxA
RKsjy24B9Zujo6RSjXckvheq17q7EDCqWICJUlJNmNkA/aZoaNiOST74ZApfX+vvFJoqoM9VX1/
LxNTliAiFJhZrt06y5mhoGI5FEx38zVI5FEx38zr0lVVUD1qWICJU5HWW27HPOsuZBNfe5lVQ90PxEuyfn
9oqqgcdSxARKgh0lrNiJpPodqyDln19b8EUkDEKUrPtDmIQsgQRIesLYYaMimXuuaeSRDp4xJ9l9gZ5QTJ4KuMnZQUyNjQXyCL0xvzNOmSe
AGG0sQESq03tRmqKhYDinDIW9S/71nXqkliEHIEkSErLOcGbcGTIqlrkB+nxJ/fee+ZNh33Zo2N1
/72mizhJEhGxmOTMktLVA1Yf9V7wUYBXVg1JECUJEviUiWeLcLyLLROTMaAc3kFhnOTMktNjfDwcL
yLLROSMaAc30BRk+a2IySS2

```
6jVu9NX+ThB51tR1MIr0DuLLqloHnAGMAK4Ebo9aVLG2Zxs01HS7m91BmIRXvsw993eCyBnnO
t5ZPcSgEmmCEO/5LOAvqroyaF34g0TmishaEVkvIjeF2H6SdzfSKiIXddrWJiIrvMf8COPsud
2b4FeTYeUz3e5amJNmneVMYqtY7pqkjpzQv+/rS3JzVFuCGFQiTRDvicgLuATxvIhkAu1dHSA
iScCdwJnAVOBSEZnaabctwJeAR08O8RYOqzvAe50YYZ8/ljIfMQtj0Rre7Fmb7aWmzznImgVUs
dxXU0u33v57Lt0H7BptIE8Q1wE3ALFWtB1KAq7s55lhgvapuVNVm4HHgvOAdVHWTqn5AN8kmq
kSgeLZLEN0MJlaQ5Zq6Wksmk5Bam2Dbyv4vXgrIPwxqtkJzfXTe3/S7SBPE8cBaVa0RkSuAW4
Dabo4ZDWwNel3mrYuUX0SWisg7InJ+D47rufEnwt5tsHN9l7sV5bje1DZon0lI2z6C9pa+TTH
albzJgLo5qs2gEGmCuBuoF5Ejge8CG4CHohaVM15VZwKXAb8VkYmddxxCR67wksrS6urr3Zyqe
4543vd7lboGpR23Yb5OQ+rsHdWf5gelHrZhpsIg0QbSqquKKiO5Q1TuBzG6OKQfGBr0e462Li
KqWe88bgUXAQX+1qnqvqs5U1Zn5+fmRvvXBcidCRgFserPr3YYPY1iSj8o6u4MwCah8OaTnQv
bY7vftjZETQZIsQQwikSaIPSJyM655679ExIerh+jKEmCSiJSIyDBgHhBRayQRGSEiqd5yLx
isCrCWHsuwnoIEaEg228D9pnEVLHcDdAXjQpqgORhMLLEKqr7W0uDa40ZBZEmiEuAJlx/iCrc
3cAvuzpAVVuBG4DngdXAk6q6UkRuE5FzAzRkloiUAV8A7hGRld7hU4C3RHnvmh0tGH1KXEfF
xB7q2Dnhi53K8j2WyW1STzN9VC9OnrFSwH5h1lv6r5qb3P9VV7/NTx4Ltw+Hp65LiqnSo5kJ1
WtEpFHgFkicjawWFW7rYNQ1QXAgk7rbg1aXoJLNp2PewuYHkls/aZ4tnve9DrkHRp2t6JsP+9
tsfFkTIKp+gC0PfoJIm8yfPycG9KjPyYjGgpUYfcnsHGRe3zy2v4xrQ6ZBrO+AhM/E5VTR5Qg
RORi3B3DIlwHud+LyI2q+lRUooqH3EPduPWb34SZ4VvwFmSnUVVbSXu74vNF6VbcmFiLdgV1Q
H4ptLfCro37K63NwfbthE9e9ZLCK1Czxa3PGg2lZ8GEU6DkZMgcFdUwIkoQwA9wfSC2A4hIPr
AQSJwE0bkeIkw5bFGO6yy3c18z+ZmpMQ7SmCipWO46jGYVRvc8gTGZqtdaggjW0gBb3oYNr7i
kUPWBW5+aBSUnwQnfdEkh99Do1RGFEGmC8AWSg2cniTgSbPFs+Ohp9+0m96BWtcD+znKVtQ2W
IEziKF8W/bsHsEH7gjXthTX/gg//6oqN2prAlwJjj4NP3+ISQtFRkBTpx3T/i/TMz4nI88Bj3
utL6FS3kBCC+0OESRCFgalHaxs54qDaE2MGocY613ntiIujf67UDNeMdqhWVLe1uLuED590ya
Gl3l2PQD3C+ONh2PB4R9kh0krqG0XkQlxzU4B7VfXZ6IUVJ7mHwvBDXDHTMV8KuUthjg23YRJ
M5fvuub/moO5O3hAbk0kVypbAB0+6QUHrd4I/B464xCXlsZ8C38AskIn43kVVnwaejmIs8ddR
D/Fm2HqIkemus1yF9aY2iaIiMMT3jNicL78Ulr4F7e0D9oOxX1R/7O4UPvyr66eQ7IfJc11SO
PR01y9kgOsyQYjIHiBUzzEBVFWzohJVPBXPdlk+TD2EzyeMyk61OwiTOCqWQ/Y4GJ4Xm/PlTY
bWBqjdCiPGx+acsbKnytVjfvAkVK4A8blK5pP+E6acA/7B9ZHZZYJQ1e6G00g8HfUQb3RZD2E
TB5mE8PEL7lE6N3bnDLRe2vFxYiSI+l2ub8cHT7jKZm2HwiPhjJ/B4RdGv2VYFMWvenygypsU
VA9xVchdCrP9LLPOcmYwU4V37oIXboFRh8MZ/xW7c3fMT70WJp0eu/P2h73Vrs6mcoX3/D7Ub
HbbcsbDnO/C9C8kTBNeSxCdiUDxiV32hyjMTmNbbZV1ljODU1sL/Ou7sOxBV+xxwT2xbTmTPh
LS8wZ2RbUq1FXsTwKBx56K/fuMnOCGRp95tZsyYMysmPZRiAVLEKEUz4aVz7ru7SGmXizM9tP
c1m6d5czgU78L/nqVKwqZ813X3j4eFcX5pQNnVNfAUBYHJIMPoH6H2y4+V29SMscVHRUeCQXT
wZ8d37hjwBJEKMH1ECESxP55IRotQZjBY8d6ePRiVzl8wT1w5Lz4xZI32X0J62LUgqhpa4Wq9
2HzW7D5bdeDuWGX2+ZLhkOmuDqZwhkuGYyaNqD6JsSSJYhQ8ibD8HyXII7+4kGbizo6yzUwfU
zif4swCWDjq/DkF8GXBF+c7zpkxVN+KTTWwL5qyDgkuudqaYCypS4RbH4Tti6Bln1u24gSKD3
TFQ8VzYBDpkKyfekLsAQRiogrUwxTDxG4g7CWTGZQWPonWPA9yJ0Elz0OI4rjHdGBs8v1d4Jo
qIGti10y2PK2G0akvQUQdzcw4zKXIMedMKhbGMWCJYhwimfDqr+5Di4jSw7YlDt8GClJYgnCD
Gztba6V0jt3uY5ZFz0wcNrh5wWauq51Zft90bDbDXC3+W1XbLTtI0BdcVHR0XD8N1wyGHccpI
3oa+RDiiWIcA6ohzgwQfh84k0cZL2pzQDVWAdPXwPrXoDjvu6ascZx0LeDZBXBsMy+V1Rvfhu
euMJVKKeku6KiU252dwijZ8Kw9P6Jd4gaQH8xA0x+qWuKt+kNOPrKgzYXZqVRYXcQpj9sedf1
LB59DKT2Q9/U3ZvhsXnuw/fs38DML/f9PfubiOtz1JcE8d6Drrluzji45C8uOdgkRP3KEkQ43
cwPUZjjZ/mWmjgFZxJCQw08dzO8/6h7LT5XRj72OO9xrOt81ZNWPlvehccvc2XuVz7jhoweqP
JLXdFQT7W1wPM/gMX3wMRT4aL7regoSixBdKWLeojA3NTWWc70yvqF8Pf/gL3b4KQb3YieZYt
h67vw/uOw5D63X8YolygCSbH3S+MAABwQSURBVKPwyPCtbN5/Aubf4IaPvuwJ9w19IMsvhfcf
g8bayPsUBPfjOP4GOO0nA6voLMHYle1KYJ7qzW8elCAKs1xnuV31zeRlWLM4E6GmPa7i+L0/u
4raeQ+7oiWASae55/Y22L7KtcTZ6iWN1f9w25KGufb5wUljeD688l/w+q9c3dnFD7neygNdR0
X1Ohgzs/v9t62Cxy91PZzP/wPMuDS68RlLEF3KPwzSc10x01FXHLCpMMf1haiqbbQEYSLzyWv
w9+uhZqubQvLTP4AU/8H7+ZJcT92C6TDrGrdu7/b9yWLrYlj8R3j7DrctPdfNMXD0F+GsXw2K
YaSBoKaua7pPEGsWwDPXug5rX1oAX1oAY2dFPz5jCaJLXlfHp9sOc2ciPUtaCjTf7LgFC9rUQ/3bWn6gd4kTBk99Y2lwbn
mPuhZQJiYsQXSneA6s+rsbsTGjJjcZq9SaaHxDH3x5BP+NdJZhBHV1sW8y2EfSnUlJAoWtvHqfoG6sCi/P/whOuqJ+bmdAWY8UCZm6k+08b8tLNVqDsJS0RTycr+M/cVrV8UiSKr4uVT+iTnDxrwNDKhuqyHLPKRCKJZNb3E/k62fNUiSafIQYJD1qRmDu8U6yWmgO2wl5P5cpHGpsVXL3HPnjHpY4wrr3CrlmMsOlsdgIrXmCD2DeQY6k9S9pY+W3iLqpNfCAHjeusV7KZYf68VSLSzmGTpLyIpXI5X+LXd9xXOjQ4Gf/YzE2A7zHDN+dTaqtXF6IwHCY2FJ4JhW1LZIMrDPecT7G0AZPyTTDufbH4hxnbT7ajLMuADrGWdd7FcE9Xx7D69/zXNmMjNJtuzpeDGnp0dT1MaNqZjYVyV2vbRryhGN0IvnQPnP1jKRRCL5jAVwiUOe8qGK0RFuvUjYz+V8aX7vJZbcMuCyZKhmnl1kqr2fEMBUGy9gsZWm6X/oiKN6oOgzRzX8WHEDyS4rtSHD8bcXM+mzxCnT2B2ObPr41xMBLTfVNVTvVwOOf0wM5BLJ0e9wFqF/jN3ENYhcHs3t2lBB81o+BD5XuPQdt0K0v/MWcxUmTGmI4pF1Ye0UZRkeoy6WQMcDGz91T1Z5YBTUdIE1f99hETCJ4b4yQ4aJX0Xro4bgZ6AHbMnJqnGnumeH2wFZdUFjHDiIPUHO4/DMinPdxV7d8x2/gJuQKBf7GBIqvoqg+fOoHK+/oSZTeovrPEyYN40cMcaZYXYySQr5BSs7/1h+/QaSfukdjpHPY9MKrxWRVmnYhuBtZdtKz9SLEAYMPpPiA/KV8FwJ9J7M+Ul45jaWyGjBNJtFpa/IBqkjdpKOHHVwJqPsR6UQwU/C3SQ5SSiqWm6H+wPwQ1DYPH3FEnaKpA+FWnCEVL89F8ADH3SUF+rjSl8EAdA8/9xa7gZdh9Yp+mmW/MDHhIyYJdnrPwXLdRKUGCvGLuEEa3PTLy3hiG3vcxpGrd+yxJGjq/Ec6hRA1rDwPAznr4uJEzcbw6PWFL6G09l3Grl7b1tN79F+8X6fdsHtfxAP5D/F0JmATKsG64gq2Nxe2EMr82Ddf+9oJSOyuTHGeLYKtIbIwEyLf70BvXWh6s00MCfWwq9uvFGVR3bQt+mn7TWrUYCe+HpBOnK3w6wUbQrjh6Pf1oCmhThbFmSe91F/wiN77gzq2Vzdm/+8N+6tb9IG6V5O3nPpw/ldzqXXTGp+rEVoLeZXK7W3XXJ8kb3tJW/TM6S4XNW+HgrFKcB0Fe8/sdg87IBqH7YqS4DbgD0dPsxD7wuXHE1sY8xn86sDG4/LAPg+YOqmr76mB4kwdEyfM9vOfFfbjjKwJZZ1ObHIpAfa4GftttEMQDfVaNoKZtMpgF2m8/eBaVaZuABMEmHYQBFoj5vpV27t4JWW2jf3rK4vBtB7IP0mcYTvAVRj0pGL7q3N6W4YmSO+fbgrm+x+ZmLaVxfO4dLr3SF3ch3tI8C1R9B1qiU9+qx4kWa8Hv92XYaEIYNKP3YXoLWCswDoCspiqoxYHs/nfEbM/bF/lF8nnyfaigjLm7X2eN+1NChdfo8QA9zzTZx/K5Gip6lI7Y9Ljyt71Dqb1YLFGxhuoJudtX5W4D5Yxj1h4MmTvEdeZi69M06Lc/bHYpjgcoY97MqF3xYYGM7nvO8Eex7Sbi+2Lzy3Nh6rFv1SUmRq6ezYdVVdGZbK8Lhd4HawfcmwEH69UhHwPPnGMpvbZpvsVzyf7C9sG77FpE3RtyAnF2ds1OK0pL1zi41jJhVOpDwELSzSDzqpvLSfvXHOjSoT4pWcepwRs1FoWowYdoTWWwd4LvWVkqbjmmVWuDnMcqnSELLQzZNXnphDmhLdSzKgtLqu7Lj3eN23Ak9v4FZs8zhuZ2+VYVNyXgiRFTiRwDVJNBfnLXPUDMnJrBcvJfHUk6HNJf3UWe1DhXVa8X34A8jxfHZrb23DjxxXhyXnqXBHNJYVl5Rtr3FTeZuuT7ZfX6R5RnHj2kvdDRXgnWNmQhJ9MX7jkgPSh0TnSMPNBdgyvYEj8gZ9c3vBJ3Xhkf0nZkCumuQFbXb4xX9kxX3fvAjk0B91FNFqbG2sVLR8JZ2SpcJHOFd2jU0PJeaVkVlsiLqeqR3KjPXD7LPqtQ4IWkO2KM1HObqIXu0IE5bF2XHyZqC2oFWoDsOIdtTkHa0GVCMfePp6mFF0E6VJ4jU8JTxWD2HnLrJKB7MyYwMyCnZeOIgN1yrKEW3AUJqpKPsXE2ZkSQfSS0UT+sKWh8bGF2SVENkt1m4bzJXFLV9xm6ZvpfbM7QXXNbMVvNUm3e1L5nNq03HuPwQn5UNvXjkp7d0rXL2Xd8P/qE2n2Iqkvy0z5Lq7fkpxhfu6gR40RJKjO8lDm33NhyhJQy7nQB1M3L3zF1SKTDfW2hPOGf3VV6sZN8+i6WvGr3uPWUSqLNw5xL4xLyuA6nP9rJr09bKSa3f96WJPqEM6pMnTL3Uq0MJfr/dGx7k+h5Dkl8F6pYw3DzW/nAqUmvPZbFf4nTumXb4W/wh3PvbjE8M3m9N9VxZ3r6PUhF1yL3ZpPNMLkyK+lQwa+BwWL3k4N3TMfqwKyDIiQlRi7eXm5o0d3HD3k8Z/jGEFQSHVFtVoiTEgr3c/Tb3e9RyQ0GcdXaR6FbWhJBVSGnT7oV6tMz8sSffU2CEBJuAt2zP6FG9zXOxYbQJkgdJFKQJfyzLZqFtEYYoUTmHPvWLTHZVv6BgFCMZ+jbi8wGwdBdO9PwnIgqfGwDjQm1CSl8bVIeqvXrBfVoABGqLYJrVfNBO8sjElkNLfP56bIgHLWcRqXXQOlKNe4FzqPBqTk4ZwDyLY3tSItsYKOc6z9ec6YHefdXR3M9QPMY0nlbeQ6JUhU2GrA1L9pTvfF0zoLo/VjI5kQc6BvN3Xh1BDVqJP8sm2Tp5V4PHgmuE1Pf4d8w6+VIqGfVoOz0eA1EQOSVNZ+a0iJH1WdRX+EQ2D1QHkpnFgMzv1JkK/XOzyTo3ABjPGh2NUsNijIB7K+bmy5nOMVWGGiBl9Kh9khUhxuhxfcbX7Sy7xN2V+l5IixchP0ecMNTO+bvC9+QAqrKmSYKebIDt7d01L+YnXkFJ8Y7GJkSx0Ms74D3nqNAnkPl4mNO92ayJThCJjF72cWvCF+xwjLZsA9KA9J1IqkZMjDcIPIHyS2Pmyzt97xnd3hbK3dHu1O1uE70ctuMfOTjWkwoN2DKgtfjAWkUvRFVBsA6TQR1R6rJrhnRAmPb4qNlf7eKCJNTMIRXCA2RtEdPGlOeoOpqfxlZgl1q7ZB0+vvt4KaTj8EC1f3Mh4G0zUZ8xmKGvDPt7X9l0GKxv+JBdSDzTHAmVeT4yHp8rwoIzkqrK2sEVh30I2bf9I3mu3tEqXLOJ0JcRDGwXGsVqwu8a5KoRvZmuXa62sxl7n0J5qoSkEUiffgb1mIYRFN7FmVuuNWmKQfDqMGaIdfH4qwbY1qU6YcqvgfDFWmfQrBzqAhbhs6KgZe2P8c8OGpl5fIYvZsaCT5mClj9ZtcL7LwmXqxoC9PVUsGNXr0Dtt0vfLnJBDmrJ74G05rHWvTK3vKmrjzGOPzfhZyMTT9L4zgQtH8uEb56oiiT4lIZCcAfwOOn2UaVtLDemjXSJDHzWIJDQmZn4iw5E6uvADFG8pWdaKo1ISfD7Ehgvsb7CIxNqbVgQHNX7BY7U3gHBP53ZuI9rbYECl9DQqyKdGHHZ2WJc6Bx94Iz6KKv2y8g3hNB2Wq7Z6oBSeFRoGa1qsNBojfDzZLt0JpR6qLr7jMIJCjJNmdJmCyBxcCzeyeNWsFvXqV2Mn1oqYFpG9xQbZdM7MADlT8eqmxtnpcaPk8zO5xXhpJ+4DFWpPdEytcpNZ2b8HmOn7lgWw5qdTvpr1uNjnVXeJQ5mJGjHjy2EMGVb5r4lNXZ/mk+oaFvzzHGnBwbkeDsuBgjYTUqPAYI5N9cfoK4xM5ZFMKH8RSh/WSl0oR0Oa5oZW2Swpls8nqSx7t6DVhE2I1V4g5zkn7n0mn2GIO01y5XbGgm0+Wa/GZGdzwkFmoPsvRNRtkNOMZUP9a5F4mfNlB9NyvSQf/ELYZtdZ5TmaG76b21W7xYRuJEq9bbdbP8eV8c5nZ4nUBrjqjwAWj0fNP0vYTWKxV+nCrHlADyvwF85JL2bFPHNQf05u3pbwT3qXiUCoiTaNcDHz9yIxM7nH0DDOTAurBhqR1SAvkuXGbqg4B7M6Dl8CDnqLTjhlL1h6XJ1gmN9jsXHPImyHz54Pbb8vTaiuphqlPaj3yR45fZpGE8XZQYa4Pl8ITBOlbX64SWnNE6dmEFJ3xFcD2LPnlZzugfZbSkCvY5oxSQw8dNsAzqK4iO2kBwr8tE7D5Wq8MDzcUXKbgVlyvuKeyK9O7a0b94aiO4lj7Fwv9aOZJ5e2lRo2xVLSF3bpvUa3X2nm0Xp9BlnXMeSbOU8MtIk3z9jxFnw0SC8uAHo6XiRFFm85tMgPKaEfvTq1gFPMrQhjXlX5O6HB3bwlkcPjKuq9hpHrbeswLpzq2bcQK2L9HnlyBu4HSv4sKMqlNTn5QyDkEhI9v2RsOQVyMLsqn9eWnDPL7XHMY0UB3KzDuKdRxJUHaOuGKzDu6cEpqXyLWdHqXIxRRNpkmhpRPg0NrCMvkfKpu8HHR2Mr0Y4aC2lBWpDY88K4p6ojRr3g5P6rAAFGXb1qA4Fj3uWs2JH3mgjvHhZ2d5H6uuf6aqgl1nnQrd7xHqoLcV+E2f7Io/NsCRPakiDaNrnjQ0YxQELw7kRTKq5XRFdJBWCYbOo7AQ2HoQgsk1x5G37QVbk5dzZgLKxlHTvXg3MaHYRaaJQI6w9ZOGCS6h9VMBJw1E0xoScIB3gvwi6VuCEnM5ZoYeYAbNEGdmPzMN8gS7yuaKs6KpbfqnlMMuwE5IyF8RgU14tGlU4WYJ1lYWpQkRH5JFlwgwG+E9lH44VR9BXPk3Gv2LNqILP8Sr8S2VCoe6X0s7T35WsUdXX5uCrwjG+OK0X91FfUE8uamgVKJx5GaDcGL80jT0SyJ53qwKzDnYqK0C8I3b5dZNaPVKC63Q4NTOzeGrC8T6pZi9t8O5N0V96TjXSG5zZvCNqxPajZ+j3gYhJx3X9AnoHmsH4Q36GhVVJ3nDvQ7y1iZg2VCwq3xEUFLHw0JmEu5zhBs7EMH0ogZqKJKTm9WvoTvAfkT1FcLOZXOfspXmN7tZIv9z7BLBPXI4e0o0gQZ4MtFlrmDGxMORlMOZMzE8x9AaMRmYuJJ1jUk3RmDaNgWINDbaUkvRdeLqqrRe+ZBY3ZDc7Sl0aKzTD2bVIVSVwGD9jQIYg8PzsZWtoNX4DmiNoMO4GLnCJNC3aDIvSKz9ZfJ71aaxIMmh2mmmeZN8h9l9JgvbSXk5Y4Q5oSkDWwKFpVyYe2g1WpwUB8DdHfl3Q2jZvF1HUJLcPk5rEHdfkuCuPxYGDLgkbV0PlQ3hxIZ0KjbT0u9FkyqR1tRIZfqFqqx4eCBPrjlrDhB7hTnM7Z+YdMDHiwGTI4NZUMbRj2hvLCkBYR6mSqKCvv1fFuYWhQiYZQnGfD1qXv8jA5t6S+FoZ6xg+vsj6J8sqLZ8HGhEcQzDRkXkPXNx0Cvtz0F8KtGXx64G5iLuVtnCdvO3gQUPkb9dRnr6gRiSaKHJCoAkvT4Yt2xgvXE0oZvoNHG0YUf3WZT2+ZoPqKAUxgbaLQ2BeCU2mYKwH6BLBSsxmh+YfP0yYVFlRNtFYJRlvFYrO0jUrN64Ohl3/LjJgzTtSC7c3UYuTOAxfPtUYnInqhgpk3gBpFWUcwkJ8o0X5P+Xsz3gu4oCbdpQK6XeRHF3jwWb+UpcrZ6VHPaE9s3g1n1UoY1LfXYdaEDWB82jXdPIcpVfGJKBvJVD26EQxdRoFFiRXGZ17nWw3lq8Vp9PSSz2udNPuI9BFzRWbKQ36U5gTNEz94v6BqzqT1h5PPkdJ1y8gO7oMIQgk2YFJXYNbmDX1uK8sbYcXHnnX6K1PEYRV6fVsOOAGxSjoFONwrbaPiu0TSlyA5TLhwsJLCP8lsAg
```

```
gRmSsia0VkvYjcFGL7SSKyTERaReSiTtuuEpF13iP0zD2x4PPtH5epkyKbWc4Ea22Gl34K953
uWitd8TSc89v+6fyWqPInuxnYdq53r5c9BH8+212zryyEyWfEN74hLmpFTCKSBNwJnA6UAUtE
ZL6qrgrabQvwJeB7nY4dCfwImImbE/s979j4TONWPAdWz3c9VIOmSCzI9rNiq3WWM7j5A/72d
TcO0IzL4bP/DWk58Y5q4As0dd22Cpb9xev89hk3bpR1fou7aNZBHAusV9WNACLyOHAe0JEgVH
WTt62907GfBV5U1V3e9heBucBj/iE1vHJAgCrP9PPdRI6qK2C3w0FS9FlY86pqcpuf
CpY+74aNNZHIPdT3I//UdaN5rnd8GmGj+FkYDW4NelwGRtu0LdezozjuJyHXAdQDjxo3rXZSR
yD8M0ka6DnNHXd6xOjCz3K59zeRaX4ihQdV1Ylv1d/eoXgMITL8IzvzF4OigNpCk+GHkRFcPc
d5dB/x/mfgb1GlaVe8F7gWYOXOmRu1EPp83T/XrB6wu6Jg4qNESRCJThaoP9ieFnevdt97xJ8
Ksr8BhZ9u8An3xhT/tn8nNDCjRTBDlwNig12O8dZEee0qnYxf1S1S9VTzHDXcQVA9RlLN/4iD
rLJdgVKFi2f6ksHsTSJJrjXT89S4pRHsmtKGiYHq8IzBhRDNBLAEmiUgJ7gN/HnBZhMc+D/y3
iARqqc4Abu7/EHsgeFwmL0Hsn1nOmromhPZ2KF/qJYX5ULvFfbOdcArM+S6Ufg6G58Y7SmNiJ
moJQlVbReQG3Id9EvCAqq4UkduApao6X0RmAc8ZI4BzRONqnqjpNVXeJyE9xSQbgtkCF
ddzkT3
H1EJvecFMWAnnDU62z3GDX2uxGUV39T9dSra7cDf8w8TPw6ZtdhbO1pjFDVFTrIFR1AbCg07p
bg5aX4IqPQh37APBANOPrEZ8Pxp9wQH+IQGe5KksQg0egknnjItjwirsjbKmHpFSYdDqc9mOY
/NnIh582JoEN6krqmCueA2v+CTVb3CxWuJZMFdabemCrLXcJIfDYt92tz53kRumdcAqUnGQd2
ozpxBJET3T0h3gTZgQSRBrvl1lnuQGlsdbd6QXuEnauc+uH57tkMOHTMOFkyA5582qM8ViC6I
lDprry6E1vdExWUpjt57mV1lkurlqboWzJ/juE8vdA29wk9uNPdOMhTfy0+/3Z78iYiFmC6In
AuEyb99dDFGT7aW61znJx0d4OKx6Gl26DfdWub8LoY2DOd9ydwphjB8/kOcYMQJYgeqp4tlcP
sRVyxnbMC2Gd5WJs62JYcCNUrnDTbp79G1dHZOMfGdNvLEH0VHB/iJx5HTPLWWe5GNlTBS/+C
D54HDIL4fP3uWEurOjImH5nCaKnDpkG/hw37MaR8yj0elN/smMvMCq+sSWy1mZ492549RfQ1g
yzv+M6r6VmxDsyYxKWJYie8vn2z1MN5GekcuSYbB5bvJWvzJ6Az2ffZPvduhfhuZvcGEiT57q
htHMnxjsqYxKezSjXG+NPdGPz1JYhIlx70gQ+2bGPhau3xTuyxLJzAzx6CTxykevgdtlf4bIn
LDkYEyOWIHojuD8EMHdaAaNz0rjv9U/iGFQCadoLC38Md33K3amdfht84x2bXcyYGLME0RujD
t9fDwEkJ/m4ZnYJizftYvmW+Ex6lxBU4YMn4Y6Z8MZv4PAL4T/ecxPWW3NVY2LOEkRvhJin+u
JZY8nyJ9tdRG9VrIAH5sIz10LGKLjmRbjgD5BZEO/IjBmyrJK6t4pnw9p/uXF+skeTkZrMZce
N597XNrB1Vz1jR6bHO8KBob3NDX3RsBsaarzn3dBYs39d7RY3mmp6Lpz7e5hxhUvCxpi4sgTR
W8UnuufNb8IRFwPwpROKuf+Njdz/xif8+NxpcQyuH6i6UU6b9rq5gpv3QvO+8K87kkDwh38tN
NV2fZ5hGW74kk99HU7+vnV0M2YAsQTRW6MOd0NCb3q9I0EUZPs558ginly6lf932mSy01PiHG
QEmvfBP78D1auDPvD3uQ99IpzF1ZfirkXaCPfIGOXm8fbn7F+X5i0Hr/NnW92CMQOYJYje8iU
dVA8BcO2cCTyzrJxHFm/mG6ccGqfgItTWAk9+ETa8DBNPhZGZMGy4G/Z62HD37f6g1xmuc1rw
a/uQNyYhWYLoi+LZsHYB1FVAVhEAUwqzmDMpjz+/uYlrZpeQmpwU5yDDaG+Hv18P6xfCOb+DY
66Kd0TGmAHGagL7olN/iIBr50xg+54m5q+oiENQEXrxh/DBE/CZWyw5GGNCsgTRF6MOh9Tsjv
4QAXMm5XFYQSb3vf4JqhGW48fSm7+Dt++AY6+DOd+LdzTGmAHKEkRf+JIOmqcacMNvzJnA2m1
7eG3djjgFF8aKx9zdw7QLYO7/2CioxpiwLEH0VfFs2LXB1UMEOefIIkZlpXLf6xvjFFgIHz/v
6h1KToYL7rG+BsaYLtknRF8F6iHevcd1CvMMS/bxpRNKeH3dDlZV1MUpuCBbl8CTV0HB4XDJw
5BskxsZY7pmCaKvCo6Aw86GN38L2ly6EEqVg/dJ2ebIZVgbdKMD3dz4AR9EPvhoplbgdwHm0DDcgTNU+/UmaHACxf+wvnOt/hkb23KfNcb7aJJTn36KpmoUdEb8TYaR5++7Nz7nYp+EgQ/P1CHfKRYB14uryHt4AjhDS5U
5BskxsZY7pmCaKvCo6Aw86GN38L958O21Z2bLrs2HGkD0uK/11E9Vp49Atu2IrLnwJ/VnzjMc
YMCpYg+srnc9/IL7wfdm+Ge06Cl34KLY1kp6dwyayxzH+/gsrahvjEV1sGf/m868x25bOQcUh
84jDGDDpRTRAiMldE1orIehG5KcT2FBF5wtu+UERKgXs3s/6wU0p4Y2WzPLXmUVUqwXObn8zt2bHe/hPtOVKORlDXaq7bzeHWy0LeuopdDM4FK0kCoD/eqNoI6gE+8Sz7jLVnLfaGIPJJ77Dx0WFBEsZxwW9+QuzupZwQn0vtjUZC+0IeWvBBbvxPMBRy8Ocb+uQEYh9OGz3Mx/+E8fj1t5SsqwS8n1PwpPpTzQZnvjFMPfZ7YSKbncXgXgNlBhvHU
84jDGDDpRTRAiMldE1orIehG5KcT2VBF5wtv+rogUe+uLRaRBRFZ4jz9EM84+E3HTXt6wBKZf
DK//L/zhRNj0Jl8+sYR2Vf781qbYx1W/Cx6+0A2DccXTMLIk9jEYYwatqCUIEUkC7gTOBKYCl
4rI1E67fQXYraqTgduBXwJ4+10CHAnMBf5PRJJqCgTzDXTHgLwRJV1MyvsMkc2jNo5KOM7ifDPu+5ZBOJGgBnR5CcnaSZ8a+sv4AKagk
4rI1E67XQPsVtVDgd8A/xO0wA1F8X0qfSRccLf7pt7WAn8+i7Fv3sznp2by6Ltb2N
vUGrtYmuvhsXmwayNc+igUHhG7cxtjEkI07yCOBdar6kZVbar6OAO4ndsD9TE9YQdsB1xzw
vUGrtYmuvhsXmwayNc+igUHhG7cxtjEkI07yCOBdar6kZVbQYeB87rtM95wEPe8aeBs0REvPWPq8paL
y4mfgG2/D8TfAsof4eeU1nND8Nk8s2Rqb87e1wlNXw9bF8Pk/QslJsTmvMSahRDNBjAaCPxHL
vHUh91HVVqAWyPW2lYjIchF5VUTmhDqBiFwjIktFZKn1ft+Oz2XNSNOv/Q0wpLL3LkxQjpd+nHFwE/3S0O4Ac5A0G8VhLZnkHdxzQeU9iyvOWC+z4+aTyF7RHU1v5SNpVMgk2h/L0Yyh8TcFfG8cZDpQH9rq5iI+h+H5vtP9XFsAAQKnZMXqY3Iv3r4Np6Y1oGfJPHVWlCMqyAZqFP4nEELz9dBv44XM6IMCg2VNFqqZ8JQ0TZnv4wXsdQxkEzlE2F8UXfNqKbGbMHtMQxoozzKUcmV9dlTzGHFi1PEzsZW9Ykw1Yb9UUGmGw16u3Y3
vHUh91HVVqAWyPW2lYjIchF5VUTmhDqBiFwnIktFZGl1dXX/Rt9Xw4bDZ38G175MStYo7hn2G
w59+au01pRH97yq8I9vwcfPwed+BdPOj+75jDEJEJa6BWUlcC41T1KOA7wKMiclDwTq01d6nqTFpX
w59+au01pRH97yq8I9vwcfPwed+BdPOj+75jDEJa6BWUlcC41T1KOA7wKMiclDTG1W9V1Vnqu
rM/Pz8mAcZkaKj4NpXWDf9exzXtgy941hY+ic3FlI0vPQTWPEwnHwTzLomOucwxgwJ0UwQ5cD
YoNdjvHUh9xGRZCAb2KmqTaq6E0BV3wM2AJOjGGt0JaUw8YJb+Mrw37OKCfDPb8OfPwc71vXv
ed6+y03VeczVcMpBbQKMMaZHopkglgCTRKRERIYB84dZnfaZDzjP6+D84Dr3Nfc3ACRAIGPsbEPMoZVHIW+eQp4ysxnnQ4+GR7DFkKGLMuQPAOCMWUCbUGRLFdKiN1Z2y8Q91ImFWJHkMHqeddRgw+SMYIInEB20u7y2pbPQIxIOfsd+RQbHEo9mJnEcS6vZkJ3g+Iro3xpwZCP1ceZR9OSxgGdvCZcRbhM7xLCVg+ijGmWi2ciUBUjOgh1d9h8VeuB9
ed6+y03VeczVcMpBbQKMMaZHopkglgCTRKRERIYB84D5nfaZDwRGirsIeFlVVUTyvUpuRGQCM
AkYQF2Se87nE848+QTO2/t9Nhx/O2xfCXefAK/9Elqb+36CD/4Kz98MU85xjxjxL2t0zkmj1Uf7N59QJBvh8ZSOHy0tgu9KJvJ+bP2V+AA7Oej5Wcz1EtSjVqK5dvufLG48NjhsPE8zo69a7pSaHUAXNFkIoOsGIi76Y3RbOFkJUSVAeg1PQFDUVIkQzKGtyvxE4RrY+m1w6rUgCpuFwg6obmLJhUtzEzAmNm9Wpm5DlwQ2JhucCgM6CE6Bt0fHVnDbwaNxEiyF99+pNIfDWMY4ffY7zUkvJpvyIU2QNpf6+ngHNdOJQ7uzXcNqDKHoKhI6u5Jx6GXpRvXj6qoQ8hDB3PzDxsKnqIOtUyvaHWEvWhQEE7MEVhzCyEKP7Rhg5ObR1u2iOnMm0RoUXMyRpoiCvWNWpgcp4R2KvRkwcWcmOD5EcWR
AkYQF2Se87nE848+QTO2/t9Nhx/O2xfCXefAK/9Elqb+36CD/4Kz98MU85xRUsJUJVjjImvqA
33raqtInID8DyQBDygqitF5DZgqarOB+A3A11R1V7RijZULjx7
Dr174mJ9XzeS+65fAc9+Hl/8LPnrGVWwnp0JSqnsOPJLCLAe/3r4K/vY1GD8bPn+fGyPKGGP6
SAbkaKO9MHPmTF26dGm8w/AXbxp6X9A8MqjL5bdf4jbTx+v3xPtDwmPVZx5b5Zg03Qdxs5ScTFSSfI2+9TUcAXzqJXC6Yz5wGPPHCV6cnkB7RKBlBhbl5H0NrYGCYN/qvw+dnZ4OCMQ2yqCsG7GLPZUmCoHsxDPJgCb+ik1HkVpTWTqJhiE57dngZ+mqmwkVWoFmYtYkJRqK/HrtAjQkNwzrc/Ahj98pWM6JWg7BUJBG2tRBf7c/IdjRLcv6kYQTwFTb1lPH6SfVzxDaRJAxoz9NtCW0g6xpPHEM8Jw81WcqT6zjoGWUNHumNshq9Z+/b11Qj+qlA7aBxtVRQ6QQ97szytsJmHCvdNaBw2ecpK1Is0U0CFYWgcHLGIpE0+U9rEMCqI8aaUT7pLyPVSdjKiz3aGmZb+Xq1fLZYbO/ZgRkIvGXwKlwh07YRSWdrkAWTNpdqbwoqEkR22wzjQ39l3YSt52cuY+Sb+cIl8nmqkv/4oAcnZkB5oFJDSeGGttd6pP4QNm5VJJI+c3DvqC2XFjEtYiYdNkDyThJEGhxMpnbKtYh1I0KGgvPEGpMuaNcvMzXtZXY+Tg4VM7BaHs+A/CRUc3YZ9WyUjyZd1ssERhgIJdeDBkxZ/bxzvr7WPzvIDbRVr6xL/DgxPgvDDlVUr6E7xgSqzjTGdxrHhO5WkklwQ9vn+yDuYzCUTWrqF/lD6lVgrLqz5sZZlvUB2+Xtcx4Qtx0NzqwDaVYYnNMqOlZsECGcMdH3BJsbH3tbERNU3JsbfzRIU8kdkLwl8mRPzuPHOH3l/ASX9oeCyzlOOGJrDUUhJGl4+Si0tVRV0TZS2dQ+Dh2rrgtYsxCJ53O31yw0cF/8mp5XWj16x5CNVHIC1fm2WHC+UptlJsCzhqHjcY/dRvWjpt5Oz1/9ByMp5UOPNgYf2nSHK8JpNNdt0Bzh5R+Zpn/yd5MmF08bk7zvt3wV1aJeFnWYpz1BdjHi5WdiQRI8z5yP+KnKcdoY5gr1DC/0/5P/OUjmT+jZ7IpjFKbiUOI2V3b6JINKUWuLGNzXJ8vYdj/rhcQgl3CObvOebTR0BQf+bmsbvyBh/Hnp9ll56k73Vkw5OzT8lebuC3WhD+Q7oUm93GtEz2jZCYdqkj2Ho6xNkZ1Wo+iJx9T5vaqxBKx8jr0WxIhq+WoSItRq11RTV7Jt3JTwiNcs6/T1VFrJYgCrMRV+8QEBFu8vXNNEEG6k0F3uRnrw/tt+6+HP8B5wj5WbKRkvGJM/fMtq78OxZK1jCPaVlRXP1a19wMoZfhaS1N9Nf5OXlFXqwbH6T+QAkmTb1pSyFmVSfP3Iu9tjEHpDRWHBtHg5pAJYNqu2jvqQRTbxOn+bBF4lr0GdaAtmTl/5hZFlMZUhL9yxWMzdY/sYyd/qDNJzXPsKIcqoTk/7PQ9wzwVhr2cQsBQOq6mK8kA8HyOfd7gdrvbCpaGSAs3KqscQ5K46pv+IVu3TW5DGGfUjmNUI1WeoAeWiy8p7BdBNE5AhK7VS/6AqRCRyDPtS1TrZDM5WHPc5dfgtiLiLdV6/1oVEkmWW5G1B0HPyYM4bH+jnHGgeyXMhhzg4Ks39oM5cW9a58bCQiGJ3FsNKsEAUi5SUyZfH52rt/Ztq59cvy+xgD7Y9Odms38ukJF2VsO1VyQzE9LALBc3zZe1qtJ8YSQCvVHnRFWt5WqtxACwaRnFvERhpuf1S7i6Syc3SdQ3kx4aNRFCJ6TGPFWPqaAfwrz/OfqFgfZbZd37lPIjyRtw5aRr+Ja8yYDwBYFq+uQ5BW+EVn2lAT6JlFxp
```

```
8Q7FGGNCsgQRJ7OKR3DkmGzuf+MT2toToyWZMSaxWIKIExHh2pMm8MmOfSxcvS3e4RhjzEEsQ
cTR3GkFjBmRFv8Z54wxJgRLEHGUnOTjyyeWsGTTbv79YSWJ0mnRGJMYLEHE2SWzxjI+N52vP7
KMz9/9Fi+t3maJwhgzIFiCiLPhqck8/+2T+On5h7O9rolrHlzKWb97g39+UGGV18aYuLKxmAa
QlrZ2/r6igrsWrWdj9T4m5A/n6ydP5PyjRpOSZLncGNP/uhqLyRLEANTWrjz3URV3vLKe1ZV1
jM5J42unTOQLx4zBn2IjtRpj+o8liEFKVXll7XbueHk9y7bUcEhmKtfOmcBlx41jeKoNo2WM6
TtLEIOcqvL2xp3c8fJ63tqwkxHpKVx9YglXnVBMdlpKvMMzxgxiliASyLItu7nz5fW8tGY7Ga
nJXHn8eK6ZXXLwqLDGGBMBSxAJaFVFHXcuWs+CDytJTfZx6mGjOLk0n5Mn5zMqyx/v8Iwxg4Q
liAS2oXov97/xCQtXbWP7niYAphRmcYqXLI4ZP8JaQBljwrIEMQSoKqsr97Do4+28uraa9zbv
prVdyUxN5sRD8zi5NJ9TSvMpzE6Ld6jGmAHEEsQQVNfYwlvrd/Dqx9UsWltNZW0jAKWjMl2ym
JzPMcUjSE22ZrPGDGWWIIY4VeXjbXt59ePtLFpbzZJNu2hpU9KHJXHCxDxOKc1nalEWY0akkZ
+RiojEO2RjTIxYgjAH2NvUytsbdrJorUsY5TUNHdtSk32MHpHG2BHpjBmRxpiOZ7eclzHMEog
xCaSrBGG9rYagjNRkTp86itOnjkJV2bSznk927KVsdwNluxvYuquest0NfFBWw+76lgOO9af4
DkoaY0akMXlUJhPzM0jyWfIwJlFYghjiRISSvOGU5A0PuX1vUyvlHUmjviOJlNXUs3xLDbUN+
xNIWkoSU4uymD46m2lFWUwfk82h+RkkWysqE2RfUytrqvawqqKWVZV1rKrcQ7JPmFKYYyZTCLK
YUZnFYQSbpw+zjKd6iWsQkInOB/wOSgPtU9fZO21OBh4BjgJ3AJaq6ydt2M3AN0AZ8U1Wf7+p
cVsQUH3WNLWzdVc+ayj18WF7LyopaVlbUUd/cBrgiqymFLmlMH53NtNFZTB6VGZemt40tbZTX
NFBR00D57gbKa9yjsR1BQfhdl+CrLS3HO2v+M502+91Xurek8TKwOJoMI9Ptm5j8DHTk56C
lMKsmhrV1ZX1rGnqRUAERg/Mr0jYbhHJqNz0qyIs5/FpQ5CRJKAj4HTgTJgCXCpqq4K2ucbwB
Gq+jURmQdcoKqXiMhU4DHgWGA08E/gUFVt6+qclsx09+eVlAAvjc8fBRdswpCcJ0vCHTx2mEeADuHGx4TkAZcfWCnQMTw9+vQhSfmmwT52+n0MrKmpFbJxVK+oaR8OMsPObT5C1PPXEtHVPGvA5eZBkZ7n9mLDa1xSV7BUWYyzbu+uYPaDmp6FtemTU0tNptG+dOWvaHnQBCzGvrCq14LO3MN6QZbz0ZUg/6XM6RWJkzhwx4fAbSY4BqMxRpLxSEFrELqnu8TgTqtGG9r/d42s4rh7V+Ls9QBGdxLDA2aVrxFS6NGh8iWyXKFaJYzjISEkSPBQRcq0N4zvxWvWnIyiYThH6MUlu+b1ab3dgKQTNbtEp7MQHdzm/VmbUr3nQwbatkSZ0aQpeUShoXJmDBO+uwnIgJXJ3ToVbn8FOvvOhaX4YAZOJoxPpCCmH8vKyt8/Q5vqi3t/h8n0bVjX+IvcfBcmBVKLhQvaPPGOG6z/zSSjm8gB1gdVy7t4PVZNqQPi9xzo4/MJfAC3BQgb/8aEd4nPXnPCFvB9Do4WAeL2TYqfhsrKYMoSP9GmgC3DLhC8yrQEH24KHCySkn5z0UOu6s+J2KDNR98VJBW8NnNe6mJfPZHVC/bM2zEUYQ75zXFhFfnhrbgxSdjqfB5k/1bYbWeSzBi52WaYGBmPgh9VZ+zgA6P/9Y7W9e8W3Fwn7ky7O/xEYAvGW90q0uV/wx9nsFx1Dj8I3EWDNc5RQ3lqqdTaVkHYCpM0xqSjL5U/W1w8ss9ZBk4nY0LL3e8O3R1a+b0Vyf6zyfbfF4M2E5QbYM3sXI6L1FsEX8lIH5Tfkbu6ziWcWKcRRtGhO+CqNbz5qHbdYX6M5xw3WGEGOj+JYZkSG3uSMB5p6/y93X6xpPd1PuZ/nlbJrQDbf7rZm6VkC5OJi+7TQ6QjnLcZOmnbMpGx5fj8YF5UXh8DV2Pb49T+L9kMPhsd2wDEAH2vgqfHwnMDEqaGvmFaTh79MYnCVYaTQ+TUWDeRYUkCnjFvJmHxDMd6YTFH1RJGqhjKX0rdcPqKklWwNDmLQx7hxVSEA2UqRUpj1L5Qy3m1pKM0OU4RPp76LKwpnO5z5eOxFF1tjtxqNRiS5nxOJQmRb+2/Z1cjAaOM9dMyVtJHnEiX9DL3yfBVnq7y5y1/mNTnK1W5+XO0SgP
```

```
C5qlofZp9I/haiFV9wndYFYc4byf97NJ0GrFHVslAb43n9eiTeteSxfOBa2HyMa93wA2/dbbh
/BAA/rlhiPbAYmBDD2Gbjiho+AFZ4j7OArwFf8/a5AViJa5HxDnBCjK/fBO/c73txBK5hcIwC
3Old4w+BmTGOcTjuAz87aF3criEuUVUCLbhy8Gtw9VovAeuAhcBIb9+ZwH1Bx37Z+1tcD1wdw
/jW48rvA3+HgZZ9RcCCrv4WYhTfX7y/rQ9wH/qFnePzXh/0/x6L+Lz1fw78zQXtG/Pr19eHDb
VhjDEmpKFUxGSMMaYHLEEYY4wJyRKEMcaYkCxBGGOMCckShDHGmJAsQRgzAHijzP4z3nEYE8w
ShDHGmJAsQRjTAyJyhYgs9sbwv0dEkkRkr4j8Rtw8Hi+JSL637wwReSdoXoUR3vpDRWShN2Dg
MhGZ6L19hog85c3F8EisRhI2JhxLEMZESESmAJcAJ6rqDKANuBzXe3upqk4DXgV+5B3yEPB9V
T0C1/M3sP4R4E51AwaegOuJC24E328DU3E9bU+M+g9lTBeS4x2AMYPIqcAxwBLvy30abqC9dv
YPyvYw8IyIZAM5qvqqt/5B4K/e+DuVFZVAFVtBPDeb7F6Y/d4s5AVA29E/8cyJjRLEMZEToA
HVfXmA1aK/LDTfr0dv6YpaLkN+/80cWZFTMZE7iXgIhE5BDrmlh6P+z+6yNvnMuANVa0FdovI
HG/9lcCr6mYLLBOR8733SBWR9Jj+FMZEyL6hGBMhVV0lIrfgZgHz4UbwvB7YBxzrrbMd6BxG
5T3H7wEsBG42lt/JXCPiNzmvccXYvhjGBMxG83VmD4Skb2qmhHvOIzpb1bEZIwxJiS7gzDGGB
OS3UEYY4wJyRKEMcaYkCxBGGOMCckShDHGmJAsQRhjjAnp/wOOa12HhQPepwAAAABJRU5ErkJ
ggg==\n"
        },
        "metadata": {
          "needs_background": "light"
        }
      },
      {
        "output_type": "display_data",
        "data": {
          "text/plain": [
            "<Figure size 432x288 with 1 Axes>"
          ],
          "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYgAAAEWCAYAAAB8LwAVAAAABHNCSVQICAgIfAhkiAAAAAlw
SFlzAAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yL
jIsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAgAElEQVR4nO3deZwcdb3v/9d79jWZZG
YSSEIWMELCIoEQEEFAPRgW2UQEAQEXPEc5V39H7hGuCIg/D171eL0eL0eF0QPCiasURYhKAESEQU
lmIQ1kLCETBIynXX2nu1z/6iaSU2ne9KTmZ6eTH+ej0c/prrqW9WfrumuT3+/36pvycxwzjnn
EuVlOwDnnHMjkycI55xzSXmCcM4515QnCOecc0l5gnDOOZeUJwjnnHNJeYJwaZP0qKTLhrpsN
kl6W9JHMrDdZI+F05fLOmxdMruxetMldQkX9vY3UuFU8Qo1x48Oh5dEtqqjTy/eCDbMrPTzO
z2oS47Ekm6RtJTSebXSGqXdFi62zKzhX2SehmdwDF1Sehmdk7ZlZhZl1Dsf0krydJb0p6JRPbdyO
bJ4hRLjx4VJhZBfAO8LHIvIUmAQnzNmAAcKnm4pSWG+wi1NUTaZf6wDG/3p6g0q6U5CajsWN3rfrWAN+JPTXyGS8TynFhKgUT9BPZl
/0xmnyeIHCXpZEEl1kr4m6V3gB6HSd1gM+9+KaZTN/V5LkMaJ0Gb5zvAc/Jx9SiSSmSdaLPJ5ZKelvT9sOxbksOlbcLFU8/d+sX5ljg1Ibpbgm+1Skbb7nOqA
VHS45J+ImlBirjTifFbkv4Sbu+SbjHgZHWIK8wu8epZR8zZyacP7fgVqQ88T7IX6idrlxWs6mxN56d+2b6Fb3qW6YqPJJxE+V2vRJb64uIkYUjllPR4r3SbzHZVE8mYRoTiUl6lxNwaYFy7lmGOBD0W7VlikFaN7VjVsZdyJb+DJMjDB
0def5PklZL2inpx4Aiyw6S9GQY3xZJCyVVhct+A0wFfh/+h/+FqqMvpRb3iy1l8yCe1WK2HOQkrgtc9VgKbS/xPCAKi1nMwlTRJ0kOStklaK+n
zkW3fKOleSXxE++ZlSXNT7YPZcCDwOJwOwvq+DpW0JHytzzZgL+/kvRG+DrPSzzaWogMdaw/CTm
bOLn5C+S/o+krcCN/e2PcJ0DJP0P/D9slfRjIUWbSdM/8DDE8LW8C0qVSq4D0X/9g+iGhCMmsRUB5uRHmEXZp5/g
QSfh1+Fz6cCrcCP+1n/WOA1oAb4LvDkp7n4vbEw+0VbKvUbG2KIGzDSb
Bn4fYnha+X9KAeuj0ai6SDgSPDeAe6r3q2UQP8DriOYF+8AXwgWgS4OYxFnAAwT7BzC6lby3
wu0le4m6gLlz/fOA/JH0osvyvssdhdxgqeLwMFcYJPA78IXyt9wBPhKv+G3AR
cDowBvgVgM0NLvjtnlWOBdcB0YDJwt5m1h+E+/xkIE4ADPzR
448gLeBj4TTJwPtQEk/5Y8EtkeeLwM+F05fDqyNLCsBDHwQJDGsBB8+PzbwIN7ua+eDqc/DTwbKSeCA/rnUmz3HGBFsv9
h+Hx6uC8LCA6LA6eXUBlZPnNwB+/9pLNxu9BG6YhRvvbykD7h8LM/VYx2mVRgHqo5WpO6ZafZb+1/yTxwkaRa5XtoYuUNqrQvbYn5kb+/kiMxUAfdqAfoRzHWJpwv6xWG7pl3s5nqrHiJ9iEsf7OE7XlqAdaVZ24uhoKemr
rP3sp3f28P/u3R/A+3viS1LuWIJkqvD5cuCCbH7/9sWH1yByW8zM2nqeSCqT9POIcaYBeqoU
uozZN7tmTCznl+IFQMsOwnYFpkHsD5VwGnG+G5kuiUS06Tots2sGdia6rXCmO4DPh3Wdi4G7h
hAHMkkxmDR55ImSrpb0oZwXuwwsI4jZ+C+hFzI3DclSt3Wfxlwr5l1hp+T37Krmek
AgtpPMv0t25M+//s97I8DgHVhr6A4kCRNAt7uY7PfGXrHGFGkvgQ4KkCWDMb
Q9BBCZE28gzYBIwPmzN6HNBP+cHEuCm67fA9xBYMPArAIUOXkxb3ZQg4qH+XD3kWHfNyRmy0pSqmDa35TOYKC8yGlE8DcpWd7f
XYvSdhmf8MvbyTYl5WReVOBDXuIaTdhf8qHgEskvaugn+pSwmCK1dcDByWZ3Z3xz+j
6v90sok/j++tsf64Gp/SS428PylwKLoj+GXHo8QbioSoK29B2SxgM3ZPoFzWwdQfX/xrBz8f3
```

```
AxzIU4yLgTEknhG3pN7Hn78CfgR3Arexq3x5MHI8Ah0o6Lzyw/Q/6HiQrgSZgp6TJwP9MWH8z
KQ7MZrYe+Ctws6QSSUcAnyX41T1QlwKvEyTBI8PHewmawy4iaPvfX9JXJBVLqpR0bLjuL4FvS
ZqpwBGSqi1o/99AkHTyJX2G5Ikkqr/98XeChPsdSeXhe4725ywAziVIEnfsxT7IeZ4gXNQPgV
JgC/AsQQfkcLiYoD15K/D/A/cA8RRl9zpGM3sZ+BJBJ/MmYDvBAa+/dYzg4DKNvgeZvYrDzLY
AnwC+Q/B+ZwJ/iRT5JnAUQXv/IwQd2lE3A9dJ2iHp6iQvcRFBW/9G4H7gBjN7PJ3YElwG/NTM
3o0+gFuAy8JmrH8iSObvAmuAU8J1fwDcCzxG0Ifz3wT7CuDzBAf5rcChBAmtPyn3hwXXfnyMo
PnoHYL/5Scjy9cD/yCogfx54LvA9XTgODdiSLoHWG1mGa/BuNFN0m3ARjO7Ltux7Is8QbisU3
AB1jbgLeBU4AHg/Wa2IquBuX2apOnASmCOmb2V3Wj2Td7E5EaC/QhOd2wCfgT8iycHNxiSvgW
8BHzPk8Pe8xqEc865pLwG4ZxzLqlRMxhWTU2NTZ8+PdthOOfcPuX555/fYmZJx6gaNQli+vTp
LF++PNthOOfcPkXSulTLvInJOedcUp4gnHPOJeUJwjnnXFKjpg8imY6ODurq6mhrG/1jdJWUl
DBlyhQKCwuzHYpzbpQY1Qmirq6qOyspKpk+fTur72Oz7zIytW7dSV1fHjBmJd8h0zrm9k7EmJk
m3SaqXlPTeveEojz9ScFvEFyQdFVl2maQ14eOyZOuno62tjerq6lGdHAAkUV1dnRM1Jefc8Ml
kH8Svgfn9LD+NYCTLmQS3u/wZQGTo5GOBecANksbtbRCjPTn0yJX36ZwbPhlrYjKzp8LBslI5
G7gjHE75WUlVkvYnuBXmEjPbPbBiBpCUGiuStTsbrh19VtrN/WwuubG3kj1kxr+243BRuQ4sJ8K
ksKqCgOHyUFVBYXUhHOqywpoLggb8CJtLvbaG7vpCneSVNbJ43h38TnXd3dg4p/X1dUkBfu58
Jw3xf07vue/0VJ4cD3v5nR0t5FU7yTxsh+b4p39Hne0ZXb+3+/saV86tipQ77dbPZBTKbv7QX
rwnmp5u9G0pUEtQ+mTh36nTMUduzYwZ133skXv/jFAa13+umnc+edd1JVVZWhyIZHZ1c367a1
sGZzE2vrG319WL/VcSoDs9H7Dl0XDTM+HKP6dTIwVWWEDz3MZJdoMa7kJ4hNN2rtR3
```

```
lTU7r5efkFQq93vcJj7mWBeyzbY8I9dCfel++H5XwfLSsbC5DBhHHxakHydi8idBDHMzIxNO9
toVxFtLc1MHFPC/Pnz+cY3vsHFF19MRWkxG9a+SGFHA53t7YyvGsMlHzmCqvwL+eWdD0DLNio
rKmhsaNj3mpjijbsO/g0bd5/eWQdtO/quU1gGk4+G46/a9Qu3YsLAXvew86ByP7jrIvjlR+Di
e2HSnKF7X5lkBhueh5fvh1cegp3vQF4BzDgJTvwqHHwGlFfveTuJysbDzI8ED4Dubti6JtJkt
xye+i786Tsw9Xg47l/gkDOCJiuX8zxBZICFw2dsaYrzngP254MnnsDhhx/Oaaedxqc++QneP2
8uWBcVZaUs+PkPWbupif952f9HnqAwP4+f3Xwt7FjHlReewfxTP8Sk/fZj6aMPQFF5cCDN1pf
XDOINQZ9An4N+NBFshPjO3dctr4Uxk6BqatBmPmZS8It/zKRw/rTgF/BgTTsePvsYLDgffnUG
fOLX8N5TB7/doWYW1AzqlsP6v8Frj0JDHeQVwkEfgpOvCX7VD3VTWV4e1B4cPOaEt61t3QErF
8LfboF7Lw3+R8f+M8y5FErGDO3rDxWz4HPXk+Satwxue8UVCZ/J8G/h8N/7vV8drQk/usLvXl
kNnDLoOzPvxof7HmJmxrsNbcQa41RXFDNpbAmC4MDaVA/tTUF7cOn44KCZrEPVDDrboL05eHS
0BM97FJQEyaKoHArLoaAYpMG9XzNo3R758NUl//Xf3pSwoqBiIozZP/xSRb5gY8Ppyv2DGIdT
42a48xPw7ktw5g/g6MuH9/UTxRsjTT1h/0BLeFArLA+ajw49B947H0qrshNjd1dwRtSzP4N3/
gpFlUESOfYLMH7GntfPpI7W4OSEuueg7u/BPmzcFCwrKAlrm3t71pZBW8PutVoIvqfRHzJjJ+
+eRIrKd19vb7Q37/7Da2dCLbw1yd2XS6qCPqkLF+7Vy/Y33LcniCFkZmxuiFPf2EZ1eRGTxha
j1m3QFAvOuskrDJJCWfXAfy13dwbt9h1h0mhvAesKlikf8vJ59e13mfXMV/Ym8OAXWGdr3/nK
Cw7uPV+O6BelJxlU7gf5I/S2pPEmWHQFrHkMTrwaPnTd8Jz6mawZp/6VoL8JoOa9fc8wqp01N
LWnobRxRZAoXvptkDgOOSNofpr2gczvQzPY/tauRLr+78HZW93B0PeMm7GrGXLKXJh4WPK+qY
Fqb4aGTSlqxuF0S5KaSslYKB47qPxEfCe0Jal5l9Wk+O71/N1/0AnKE8Qw2dzQxuaGNmpL89i
voBG1bA0O4oVlQWIorQoOukPBDDrjYcJoAevm1TfXM+udBXu3vbLqyK/+8ANYPmHkHbgGqqsT
Hvk3+MftcMSFcNZ/Dc3BJKptZ3C2VU9C2LB815e9eGzfM4cmH7VvnWHVsAme+yUsvy349brfE
XDcF+Gwjw/NfjQLOtI3v9T3VOaWrcHyoopgn/Xuv7nJO+iHS0dbcFZdNHHs3JCkZj1ARRUJtZ
NJUDlpWE7ZzukEccghhwzLkNn1DW00NOxkUkEjpd2NwY+Jkqqg6jtUVdB+mBmrV6/2O8olYxZ
cJ/Hkt4JO30/+JvjVNxgt24LmmFcegDeXBb9uR/OppO0t8OK9Qa0itjpoVjzmc8HZUuUpTqLo
qZmm/EUePo82n9a8F6bM27UPJ8zyDvMMy9kE8dZbb1FZWUl1dXVGk8TO7VsobKmnTHFM+aisO
qgxDPUv1RTMjK1bt9LY2MiMGVluKx7JVt0ND34Jag6Gi+8LfrENRPMWWP0wvPwAvPVUUDusmh
b0HRz04dy4GM0M3ngySBRrlwTXphxxQXBqbbImma72vuvnFQS/jMdM6ttkUvNemHI0lI7Lzvv
KYVlLEJLmA/8XyAd+aWbfSVg+DbgNqAW2AZeYWV247LvAGQTjRS0Bvmz9BJssQXR0dFBXV0db
W1uKtQavu7ODvKZNdKmAvJJKVFQ+dM1IA1BSUsKUKVMoLByh/QEjxZvL4J5Lgyr9xffBfof1X
76pHl79fXA9wttPB0lh/IEw+xyYfTbs/77cHdIi9lpw5tPKu4L+q/yiftrLw+nyWq8RjDBZSR
CS8oHXgX8C6oDngIvM7JVImfuAh83sdkkfAq4ws0slHQ98D/hgWPRp4FozW5bq9ZIliOGw7YG
vMWbFrTxx+p/46LFHDPvru72w+WVY+IngzKIL7oCDTum7vPHdXUlh3V+CzuXqmUFNYfbZQado
riaFZOKNQdt8eY3vl31Qfwkikz2Q84C1ZvZmGMTdwNnAK5Eys4F/C6eXAg+E0waUE5wYUA
pszGOve6eqkYvVvWdo9h6raATZXuOyZeCh8dgnceQEsPB/O+nFwmmlPUnjnGcCg9hD44L8HSW
HCLD/4pVJcOfqb1nJUJhPEZGB95HkdcGxCmVXAeQTNUOcClZKqzewZSUuBTQQJ4sdm9moGY90
7ax+nqC3GfV2XcM0ehu52I8zYyXDFYrj30/DAP++aP+FQOPnaMCkckr34nBsBsn0O49XAjyVd
DjwFbAC6JL0HmAVMCcstkXSimf05urKkK4ErAaZOnTpsQfdauYDWwnE82TaH//QEse8pGQufu
g/++qOg8/XQc6BmZrajcm7EyGSC2AAcEHk+JZzXy8w2EtQgkFQBfNzMdkj6j6PPCsmTWFyx4F3g
/8OWH9W4FbIeiDyND7SK55C7z2KC/Unk9BWxEVxdnOtW6vFBTBB6/OdhTOjUiZPN3mOWCmpBm
SioALgYeiBSTVSL2n/FxLcEYTwDvASZIKJBUCJwEjq4nphXuhu5NlpadSW1k8LNdaOOfccMpY
gjCzTuAq4I8EB/d7zexlSddJOissdjKwSNJrwFjgJ8Lp9wGLgRcJ+lE8n5icDoAE2vpafPj/c07WvxkGvpBgSwpFsekU06EWYdnfNEtJUB6obZ6ZcOOI+3p+8GWFFOSqr4RYbqQdZ2S1HDo2Cs7jbFbDAD02x8eLTpVn6VKHaFR6sQDAvEvKLBqU7nHD0xZ4AjjIsiWuD8qOGF4zRyO6Mzfh3kMCVqbB0VuhZ4RTFhPsQTAXYjDNE9v3aNchvJKCZQmd3vVqcxmV49z88NDcMLCrCPGkp1HAtwh7I4ZWLuvYvNeQM8Ox3Gx0rVUtU1lsM4pKcUdYu7lY6GMjaYIuxWs6HAWFJC85izxFHGLKdMQHjNt8e6WNNFXqt52MvxZt8fVq60XA2lOBqzrQZx7ljhjFDDUaOrsVF6qwRTPPGiuRKNp6W27Zt4AYFeE1tkKsDWBnQo4hVwHWSTqUtnhe25qbpsq8DDotMdfy2JmqtMO6rMvvzBGkFcFfqYF4pTgzLRRTCEgqTrhdxGQmUgC0Y3FOrJ8Vx2u0WVNxODm8RfUa9KK3pQhbPbwEwlmUH3fa5e0AuJjtq9S4odMwTJgkXqJ8i2fQwHK2Tv8pdBNkRJDMxFtO0CRcmO3aaYlFtZXAx3aHAh0w5FVL4Kx+Z9i5r1A08p3RMNB6GF0Gpnh3Nj8cx38EY+GJgwIvq4GgXk0qb47E+qDIqJ9XhoDajwHJb+LXBGcSL6FzL7wl9CzMX7JkKTtlbyOeulx3Zbm5S8bCjF1AdAYvAhZGz7Glm5MZ8sMAuo+SpqsyfgN0hF2f2xRfHt8Y3XiLr9RcDh03ibZQb0FmEzZ/9RV/K31/tE1eHhN/8eNPz4zaGQFVG3lXlD0Onwsm9BBwS1cE3fDQ2xpXcsU4ixsuRkk8fuATBmX3r4AcESTAQbGmsDT1/8SrsIeNMR1ytBs4/jVGyJfHUUcm2/RH5aZA5fNwT7VhgXmZVxFXAZYaVaOqCYyFKgkN9T2QIOcm1Jj9K6vBh5wT6OXAa9Qri9BdAJq5xHZ3TbHtiDvBSIJnO3ONyv7LN8w94XNAVAvjTnu3yRbHu1pNWjPDFWQ6tOfo4uzA3RQcXyfMQ2Uo4AJ6jAMuLKR+9NF4GwkkXObZmQvxpr0cmP3MHSwuUJOB2uvJXFH07jhE1XXhVdK9CXqMGiurUMvwgPcPtuHTmK8O4FyqsZmq43yKrF42UQbB5CQVFNOd5wpbNpT0Ck9x6VY7mpzDq7W5S6YX8jXsvyVv1vuVSqI0kMj0lAqSWfzeD9nuPzTwUZ4RV3K8E1Bi5hfW+nxymzUdKN4OOzI03X6ap2XP4pE5X0Y6QPC9dOKxvDrpJ8eDkIr1yxoerNQz8CdG/qZQH4k5MFE0xEydfZzT9I3jcA1e8HpLOadcI6EjYoJA1ePI7TJr1kUD5N/Rtk5EFgtAf/1BTCJDx0vwlAfN8Vm6qoD5ji5KUqbmXEmjJk1Pyt0F1dqvBp2BDwBf38GZ/M2j1MG8pZjx2Kp8G1dq/OnOHpzR6j4gBRVnkQg6uRsCJNHzRb+v8bdQL/wdzk1y2m5PHpVz0iJ1ZtYzu/Fk6VZMI3+N8M+oIQkWcRYbPQQF7vP6EMhzvSI2WeE8UPEzMVNO+gDwuREVwxuHtWs1zGcs7TIQ4n0vHTUDiUr0d+PvRjnzFu3RPjU+BwvQ/M7QxQnKEx+DiJ5yFVjJ4s6+TWcXhdH0x8YQZEFtJj0YbBUZ7sf
```

```
WksFYA81b4PU/BH0P+amvb/AE4ZzLBZlMEJOB9ZHndeG8qFXAeeH0uUClpGrgvcAOSb+TtELS
98IaSR+SrpS0XNLyWCw2+IhfuBe6O3e770MiH4fJOZcLst1JfTVwkqQVwEnABqALKABODJcfA
xwIXJ64spndamZzzWxubW3t4CIxC4bWmDQHJia2hPXl4zA553JBJhPEBuCAyPMp4bxeZrbRzM
4zsznA18N5OwhqGyvD5qlO4AEg+RVrQ2XTKtj80h5rDxDUIMaUFFBSuFulxjnnRo1MJojngJm
SZkgqAi4EHooWkFQjqSeGa4HbIutWSeqpFnwIeCWDsQa1h/xiOPz8PRbtGWbDOedGs4wliPCX
/1XAH4FXgXvN7GVJN0k6Kyx2MvCapNeBicC3w3W7CJqXnpD0IiDgF5mKlc44vHgfHHIGlI7bY
3G/1ahzLhcUZHLjZrYYWJww7/rI9CJgUYp1lwBHZDK+Xq8thtbt/V77EBVrjHP4lKoMB+Wcc9
mV7U7qkWHFAhgzObhzXBp8HCbnXC7wBNGwEd54Et53EeTtudO5Od5Jc3uX34vaOTfqZbSJaZ9
QOg7O+RlMfX9axf0aCOdcrvAEUVgK77sw7eJ+DYRzLld4E9MA+TAbzrlc4QligDxBOOdyhSeI
AYo1xsnPE+PLirIdinPOZZQniAGqb2yjpqKIvLzktyJ1zrnRwhPEAPlV1M65XOEJYoBiTX6Rn
HMuN6SVIML7MpwRGVgvZ3kNwjmXK9I94P8U+BSwRtJ3JB2cwZhGrO5uY0tTuycI51xOSCtBmN
njZnYxwT0Z3gYel/RXSVdISn1vzlFmW0s7Xd3GhMqSbIfinHMZl3aTUXgr0MuBzwErgP9LkDC
WZCSyEcivgXDO5ZK0htqQdD9wMPAb4GNmtilcdI+k5ZkKbqTxBOGcyyXpjsX0IzNbmmyBmc0d
wnhGNB+ozzmXS9JtYpotqfcOOZLGSfpihmIasXygPudcLkk3QXzezHb0PDGz7cDnMxPSyFXfE
Ke8KJ/yYh8E1zk3+qWbIPIl9Y4tISkfyLnBiGJNfg2Ecy53pPtT+A8EHdI/D59/IZyXU2KNbZ
4gnHM5I90E8TWCpPAv4fMlwC8zEtEIFmuMc/B+ldkOwznnhkVaCcLMuoGfhY+cFWuMc8J7arI
dhnPODYt0x2KaKWmRpFckvQPmmZgAABMLSURBVNnzSGO9+ZJek7RW0jVJlk+T9ISkFyQtkzQl
YfkYSXWSfpz+W8qMto4uGto6mTDGr6J2zuWGdDupf0VQe+gETgHuABb0t0LYkf0T4DRgNnCRp
NkJxb4P3GFmRwA3ATcnLP8W8FSaMWaUXwPhnMs16SaIUjN7ApCZrTOzG4Ez9rDOPGCtmb1pZu
3A3cDZCWVmA0+G00ujyyUdDUwEHkszxozyayCcc7km3QQRD4f6XiPpKknnAhV7WGcysD7yvC6
cF7UKOC+cPheolFQdvtZ/Alf39wKSrpS0XNLyWCyW5lvZOz7MhnMu16SbIL4MlAH/AzgauAS4
bAhe/2rgJEkrgJOADUAX8EVgsZnV9beymd1qZnPNbG5tbe0QhJOaJwjnXK7Z41lMYV/CJ83sa
qAJuCLNbW8ADog8nxLO62VmGwlrEJIqgI+b2Q5J7wdODIfzqACKJDWZ2W4d3cOlvjGOBNXlOX
d9oHMuR+0xQZhZl6QT9mLbzwEzJc0gSAwXEtx0qJekGmBbeBrttcBt4WteHClzOTA3m8kBghp
EdXkRBfk5f1M951yOSPdCuRWSHgLuA5p7ZprZ71KtYGadkq4C/gjkA7eZ2cuSbgKWm9lDwMnA
zZKM4GylL+3d28i8WGOcGj+DyTmXQ9JNECXAVuBDkXkGpEwQAGa2GFicMO/6yPQiYNEetvFr4
NdpxpkxPg6Tcy7XpHsldbr9DqPWlsY4B9WWZzsM55wbNuneUe5XBDWGPszsM0Me0QhkZsQa43
4vaudcTkm3ienhyHQJwTULG4c+nJFpZ2sH7V3d3sTknMsp6TYx/Tb6XNJdwNMZiWgE8msgnHO
5aG/P2ZwJTBjKQEYyH4fJOZeL0u2DaKRvH8S7BPeIyAk+DpNzLhel28SU03fJ6alBTBjjCcI5
lzvSvR/EuZLGRp5XSTonc2GNLPWNcYoL8qgsTrdP3znn9n3p9kHcYGY7e56Y2Q7ghsyENPLEG
oOL5CRlOxTnnBs26SaIZOVy5ud0T4Jwzrlckm6CWC7pB5IOCh8/AJ7PZGAjSawx7mcwOedyTr
oJ4l+BduAegjvDtTGCB9YbarGmuHdQO+dyTrpnMTUDWR1uO1vaO7vZ1txObYUPs+Gcyy3pnsW
0RFJV5Pk4SX/MXFgjx9ZmvwbCOZeb0m1iqgnPXALAzLaTI1dS+zAbzrlclW6C6JY0teeJpOkk
Gd11NPIE4ZzLVemeqvp14GlJfwIEnAhcmbGoRpDeq6g9QTjncky6ndR/kDSXICmsAB4AWjMZ2
EhRHyaI6oqiLEfinHPDK93B+j4HfBmYAqwEjgOeoe8tSEelWGOcqrJCigvysx2Kc84Nq3T7IL
4MHAOsM7NTgDnAjv5XGR38IjnnXK5KN0G0mVkbgKRiM1sNHJy5sEaOWJMPs+Gcy03pdlLXhdd
BPAAskbQdWJe5sEaOWGOco6ZW7bmgc86NMul2Up8bTt4oaSkwFvhDxqIaIcyM+sY2r0E453LS
gG85amZ/MrOHzKx9T2UlzZf0mqS1knYbqkPSNElPSHpB0jJJU8L5R0p6RtLL4bJPDjTOodAU7
6Sto9sThHMuJ+3tPan3SFI+8BPgNGA2cJGk2QnFvg/cYWZHADcBN4fzW4BPm9mhwHzgh9GhPo
aLXyTnnMtlGUsQwDxgrZm9GdY27gbOTigzG3gynF7as9zMXjezNeH0RqAeqM1grEn1JggfqM8
5l4MymSAmA+sjz+vCeVGrgPPC6XOBSknV0QKS5gFFwBuJLyDpSknLJS2PxWJDFniPWJPfi9o5
l7symSDScTVwkqQVwEnABqCrZ6Gk/YHfAFeYWXfiymZ2q5nNNbO5tbVDX8Gob+ipQXiCcM7ln
kzeNnQDcEDk+ZRwXq+w+eg8AEkVwMd7Ro2VNAZ4BPi6mT2bwThTijXFKcwXY0sLs/HyzjmXVZ
msQTwHzJQ0Q1IRcCHwULSApBpJPTFcC9wWzi8C7ifowF6UwRj7FWuMU1NRTF6eshWCc85lTcY
ShJl1AlcBfwReBe41s5cl3STprLDYycBrkl4HJgLfDudfAHwQuFzSyvBxZKZiTSXW6FdRO+dy
VyabmDCzxcDihHnXR6YXAbvVEMxsAbAgk7GlI9YYZ1KVn8HknMtN2e6kHtF8HCbnXC7zBJFCV
7extclHcnXO5S5PEClsbY7TbX4VtXMud3mCSMGH2XDO5TpPECnsShDeSe2cy02eIFLoSRATvA
bhnMtRniBSqA8TRI13UjvncpQniBRijXEqiwsoLcrPdijOOZcVniBS8GsgnHO5zhNECj7MhnM
u13mCSGGLJwjnXI7zBJFCvScI51yO8wSRREt7J03xTk8Qzrmc5gkiiS2N7YDfSc45l9s8QSQR
a2oDfJgN51xu8wSRxK6rqH2YDedc7vIEkUS9D9TnnHOeIJKJNcbJE4wvL8p2KM45lzWeIJKIN
caprigmP0/ZDsU557LGE0QSsUa/k5xzznmCSCLWFGfCGE8Qzrnc5gkiifoGr0E451xGE4Sk+Z
Jek7RW0jVJlk+T9ISkFyQtkzQlsuwySWvCx2WZjDOqu9vY4iO5Oudc5hKEpHzgJ8BpwGzgIkm
zE4p9H7jDzI4AbgJuDtcdD9wAHAvMA26QNC5TsUbtaO2gs9s8QTjncl4maxDzgLVm9qaZtQN3
A2cnlJkNPBlOL40s/yiwxMy2mdl2YAkwP4Ox9or5NRDOOQdkNkFMBtZHnteF86JWAeeF0+cCl
```

```
ZKq01wXSVdKWi5peSwWG5Kg/Spq55wLZLuT+mrgJEkrgJOADUBXuiub2a1mNtfM5tbW1g5JQP
WNPg6Tc84BFGRw2xuAAyLPp4TzepnZRsIahKQK4ONmtkPSBuDkhHWXZTDWXt7E5JxzgUzWIJ4
DZkqaIakIuBB4KFpAUo2knhiuBW4Lp/8InCppXNg5fWo4L+NijXFKC/MpL8ofjpdzzrkRK2MJ
wsw6gasIDuyvAvea2cuSbpJ0VljsZOA1Sa8DE4Fvh+tuA75FkGSeA24K52VcLDzFVfJhNpxzu
S2TTUyY2WJgccK86yPTi4BFKda9jV01imETa4wzwZuXnHMu653UI47fi9o55wKeIBLEPEE45x
zgCaKPeGcXO1s7fBwm55zDE0QfW5raAT/F1TnnwBNEH71XUftQ38455wkiqvciuQofZsM55zx
BRPgwG845t4sniIieGkR1RVGWI3HOuezzBBERa4wzvryIwnzfLc4550fCCL+K2jnndvEEERHz
W40651wvTxAR9Q1xv0jOOedCniBCZuY1COeci/AEEWpo66S9s9sThHPOhTxBhPxOcs4515cni
JAnCOec68sTRKjnKmo/zdU55wKeIEI+DpNzzvXlCSIUa4pTVJDHmNKM3oXVOef2GZ4gQrHG4B
oISdkOxTnnRgRPECG/1ahzzvXlCSLkCcI55/rKaIKQNF/Sa5LWSromyfKpkpZKWiHpBUmnh/M
LJd0u6UVJr0q6NpNxgicI55xLlLEEISkf+AlwGjAbuEjS7IRi3mtkxWDFZnY4
cDTwBUnTMxVrR1c321rafRwm55yLyGQNYh6w1szeNLN24G7g7IQyBowJp8p4uug
9CQqUC3Nbdj5veids65qEwmiMnA+sjzunBe1I3AJZLqgMXAv4bzFwHNwCbgHeD7ZrYtU4Huug
bCE4RzzvXIdif1RcCvzWwKcDrwG0l5BLWPLmASMAP4qqQDE1eWdKWk5ZKWx2KxvQ7C70XtnHO
7y2SC2AAcEHk+JZwX9VngXgAzewYoAWqATwF/MLMOM6sH/gLMTXwBM7vVzOaa2dza2tq9DtTH
YXLOud1lMkE8B8yUNENSEUEn9EMJZd4BPgwgaRZBgoiF8z8Uzi8HjgNWZyrQngRR401MzjnXK
2MJwsw6gauAPwKvEpyt9LKkmySdFRb7KvB5SauAu4DLzcwIzn6qkPQyQaL5HaZi9RSauAu4DLzcwIzn6qkPQyQaL5lZm9kKlYY41xxp
YWUlKYn6mXcM65fU5GBx4ys8UEnc/ReddHpl8BPpBkvSaCU12Hhd9JzjnndpftTuoRwe9F7Zx
zu/MEgdcgnHMuGU8Q+DAbzjmXTM4niOZ4Jy3tXX4nOeecS5DzCaK9s5uPvW8Ss/Yfs+fCzjmX
Q3L+9mnjyov4r4vmZDsM55wbcXK+BuGccy45TxDOOeeS8gThnHMuKU8QzjnnkvIE4ZxzLilPE
M4555LyBOGccy4pTxDOOeeSUnD7hX2fpBiwbhCbqAG2DFE4meDxDY7HNzge3+CmimmVnSW3
KOmgQxWJKWm9lutzUdKTy+wfH4BsfjG5yRHl8q3sTknHMuKU8QzjnnkvIEscut2Q5gDzy+wfH
4BsfjG5yRHl9S3gfhnHMuKa9BOOecS8oThHPOuaRyKkFImi/pNUlrJV2TZPlXyyXpR+E+fEHSUcMY1xRJj0t6VdLLkr6UpMzJkpokrQy/rh+u+PL93pcYYN6Ex/q+b3ERMUF3Hq82Hhhp8G6cn/GjzE6zpUK+NnfaDHpUpzP3fPUb5j+bUnRhrgGq8Do91xS2
bR0QHOM9AJqzR36/vBtmWRfbHxeKCNFWbaiNCrB2yWwgEP1Y6FeCFpxSY/MiMRm2u4vD8fq+AJebPJDk+VmImp1frHRxejMxNtpDqGDqV2NyX9aQbOe9wWYg5WiVfHq0sHZtHzlGxKtCPM+y7kkyPsxumcpn83pC2O9qlQ2K5sJbtjkhXXuKpwzdM23kntIS3VddZKWjm6s6w0T0U3Tq3O/Fp0KXL/9+6CLVxNQ6I23j4Nj+sWXTrKl0cGXr8PO8BZRQfrm7l5SjCGlQmMwNVcONkmPbNeaRmLjFiXyaP3ChrdpgsJl6V7P0PTjaP2k15TxjPjYrvbSHV2xi4B2rqd0dV+HsJiWT8iDCCH/ixflDHWtyd5nNJfiUNKTj8Dn1i8qDHfb7BPwtW2KLLGbQYN5K9Lzc1Mvr9Wd8l4cRzu4Kpx+Tvk9LLqz9VYXAdNmgYR+J5xe4P8DZcbdhrsRFMoj14F6ffe6rlmvf1sVM3cbaf4ANzqA/1+FPAJJsB+s52mpMi3aMKJrD9mqwPQmHP0f5Cn1gE6O8/v34Y1BpUfcA0aPDh09PS2DU1Xlpz4/Qz1vwSzX38wccE5bSJ/8Ad6v/N5O5IK8xQ57tOnvfJ5o4HAf4QMjUy78hs+wKg6l3dZIcVeUC2kb9FL3PPb4xG7Yy8g3m0HebbZFWlxZ5ttdUqIvqjZDXRZ2Wm5V6Sv8PNm+Ka+PDG6YpvGgFv0R7BWVjUR/DxFQn5rKp7kIgShJ2T8EiddBSPzOE4aMuPbtTTKeJ5V6rS3Hhtc0ZK5HfDIMcAf1BdWWHU2JAXkKAnjT8pJkXfVaJ5gE30z2mEQgH5Ygs/MP8n+WoErytoP3V9AIG7k2N9L0u/ugTOlhPCRfZaNrOdUiHAKsM4BldUe/l39ie+39SFtcpBFm3XWShqUOSu6H5CFm0ELnH+4RFmQ4ZExA8HsezMLm2n4QtJQi5Axe3B2xyShUwJ3U+BqOPjZxO+Bo8yDRRFrMNKCZ8SQyeshT37yfuPtwPF6Re5Wvrfk7UhU2KvNFVvxb0qIvtlJhk7tTk/nQnvrsgKuSx4Ud3E8dwmV86D1xyOvP8v9HmRHv8Jvpg/8GMhzy9+3H3qbNc4v+4ib5SmNRlfm5Okq4LflTnhHmZQdwJhrLrz9j2vZvEMb2m/tn+mgGBaQzkgjSFK+7f0zf3PPOwT+g0xEm7+9xK+tfWHemD+PVnXs2iZ3rxB1nhQSgtHBwEIQpXAs84qeQv4cd0N4RFGDBlxzkGGU0rhFpGl99uDbeGDGPlZRXr4HTMLM0ZRfZhE79jrKYWQz6YF9SpdGV0v6yBp4ylJE7X6LHmlhn2dkl/W3e9jA5dzydfyXmFuxeBu0A4xnDPMEJ+9BsKbT0kZ1ZkxN/9EqtEsp2k5AQ7tJ1ZZJTThxd+S6YU0u9R9nVWmCCcOqjSVCbiCSVLYXSNI6ZMPB3xvJ+GXKAkQZs3JHklnlw2b+vi3HZnGoNuAAxhs5vVk6p8DDxPUQJQ0q2FAlIRRYLr0K/QLzTrQdi8NAl0Ey7HTY1l8rbXuchhDcqa27nGO3Cif9eJjKkwkUDmewa2jdvjl9TsmzPVf5fVLZ6LS7wbzDSDgHSRwWe7YjGXBz2GjLX8OE86Oy6DiXfbhmjqhZhPc2Sg9xxZtYt3bw5oZ2zBxZMi0hGqnPHyMdu+4AzMn2DGjMwFHGmZvdlx4e8S1cjq6M6NtR4iQdbp3z2E7BGIIxNSRgJe9Ow4qiRJ+Lf78ZXfY4ZmdpRKsmyFnlyB4OIRmTDQNwf3pvkImCG3WvNr9KWv1f2nSDnAbK9S7h2W97rFHCdKp4J/aXpmM/0PxxSSg6WnvJe71VNnAZdX05kHb7M12xbVOm5uvn1uV7S2VKi4fhI3kZM+ZGw1BB1jW0lWPN7s0uTE9FbtZvS5LTmFj6dmG0m33TmyFSebqdwF3gJ9jU0m2NKm2V3ttn2Bc3VbY7L33dlNrQ1HSCBHfBDxx7Nwnwnt3kQWSCLCwxfqTR9DwHDJJXe8BXbm4TBjXNdJ5Y8xOPVRvyZvELTJuzVa5cCCbBPvCK2FuI0yCDMTQL5DHdd2kVDO5YnLaTh5Wc6T6HIT7nhk+vY0TBLp4Y1Gyln3fbkpFZMEpqvGLcDVqPj8W6LLX9mNWLexnp2RhGrf4HWq2aygz81gmpR0zl2WTSsAiZA5eGn1S7wBshVsnHgWjxGN5+Ldz7W1KQhHbeNfKnAAUX2FLvvU3o7zLNwBIRGh6eWvtsQ7lZiW7MVAA2GHtFyYf/0P9mxT7pVXPjz3iNKw08RvDTgqnvfLdg0Fp2bf7Wc6Ed2b3MqUBD8B9mPeXmFCFmtOvClzSOfj2cfAIeM0s8z6qnMCNAyJVQHR6v05dgyeUG5vXRNJcPc2rQl6dwlBEuaf1mcHs+UtTqCjEF93A5sSV9mzbL5JhWa+0sKrMXFuI8v9vDQrl3i1Ea0pxPzLy/5E0c7TcyaXqZCL2EOCTWGzV9BuDdmMnBLuB9NTbf1+wj6LfJzMp2QmwFGUN5+2O4/Ln+tOcptCmXxrw8tOtSr/2sVy61U0mSxFyE8k8n0h8PqkR6NDxLz8PZJn3U3zJWe2OUZKPx4N/GMGSSvyTtgYT3OzPKlg6XA13SA2Bt7F96I9Mj5Yb6OIwd4k4i6qMDkL8Rk6uUSw1HXZlgkA3iATjywrPdRZ+0aDSnWXNrUqRWgqfKGYnUEiy/JK2G3aVfxAn8R0Cwsjw0A23CSJHP0mJD2ojEYyeYZqySu/Xq+MTxBDMx/R4OAb2Hg+/0XMe9VvAn+GZ0Yfz6O6ZZRLgHuEJe8H2JtPSa5Hox2CeUhF7zrH+0JgObqvwnr9kvpAqLXhE/+SGBkLnGHHKfJwlP9dMtHQhEVJbwZrjK3BIgefsmEU/8CmYS/OnPAugSNMG6uIH+MrpBLBh7iOVsddB2MBK+rUmzyHmH9MJLIRFMWJ0TUgqRCcK6M+/rjV7eHK8E3cgshxvcUmcpHdxUJz/pdjVQ/dqOuxVrPoXdCSbVthrNp58/lnN7+nZ0z39hFBNzOCq4WvSn5uoS6aI/lhBA0kRMUKgQf5Nbh+Y+Ov1k8tmfoY5M4vOcbaCn6rUHnNW8rChwW3Ds2/L9xj/9yy94JC/9VVrz6wvPbPT6Y4t5fjqZ+J+JJWpNEVnwuxJR1Lgu9xZ8N7TgEFKkq37mDYbCY/ZV27OwH5mCYG68Y0V59lkd9cf98rmNqGJaFq40Z2pS1U5cjEmGN2PHcbFAH9Ujo9RaQ7HhjYhD5D9zx3d+pb6R8UJDCjgQdqldcKhBMHQiTdt8pFqUB2HwCZ9b2VG2VXOnp+EPSJuX8PZJzwt+mqjdBUH5mC0eCmB+Ge2ndUYjzkt+ir9+5mZYD7yqqgbtmCVqXOdqZxsIsxYR5IY7EE4gljllhsjc7Hy0PQCpo2DkAGo7k7BE8csO8bPwZcEg4xAKGKy+9E6VLFWqvgHJOBdY0+Z4KCmKV9KE/rVJOiZHWtDbe/5eiWfeWEZSwQLmdG5+RD/mEb9zgeMvVKn8CEFSeO/3Z2Lp2Tbm7z7uTz9q+6z1GNlY3L+sUS66Lg7e7QpTOdBTb2TU5v57Xe9J1bPxqhMu4Ax9bNhRbmU0VVQg0OHsYdiVyFcW5b8EUdZEvsZMqUoxFBo1UdHq7iHe8ji3zVdSvzj7Z0y0Yrm2bW9PT5OHWEOkKTvNRl8SBwqK2oP8oRFOpeQ9v7M4rqFDmkUHe+yChI+pzJI/0LexNlQ47X5q3dKjQpmvHhEnMfTe8RpjUyq2Gg3NmpRBXiVT70T0SWqz1vyOJjKKqe+UV5Q7ZGrjqmqD/X/ZDkrhNmMhZHQnKbk6P2YZ7AqYb2G6zwdyWQ0qj+MCmpuzK7a+8K7GdLg1Vs8+oZHf5nEvOuYbXrUMQG0uqX+2C5w0v1nC4ZxZ1Eyz0fcHymS/Pcj+GnL06jDQzE0LO5IDf8Bbfy1RGUumlyyvt+Ug9YOpjGsoRoKL8sNo3SPn0Sd7bI6fzh6dJHjhXFrM9BwZYuT6sOp+0ZhkJWeUZ5jnmtoUYoXl1OObUn2YgrnV3fQdYCbd6DsKdyq06Xu2MbRYw1SrIXO2SPrvuHSIYQRlrxU5yJwj1e5mEm1AvB+Lo5R6ECgx3VGVZjdmPk/6vZu28hRKnq0MkVkG1SL5+26m6f+nm2PZs+M5q/9tmIBVXWzqbRF9aomUZlpjPVRSndcfqIB6OlY8R8kT3n8VU5CqVsGmHq3jjFbZAjXvxFt8tzyG/tN7GbSV6l0tDjh9zVpvmG3D9OvAPPkiPEuqMG97sYdP4SdYkw4yafLqI0pPVcVP1vf6zqplL0iUmyGoQxbYiCEiIvu0ljGXtBDuKUsNt7EyavcwI7Ov1IwPbsmqB4R6LCI+d2S3W5R8eWEQX5j4JCc7wZpE4Z7t6Sf2rP/AQS4gE6fQvqBdn7GYmz91CR9BNB0QT3ULEB8AqmT07ZTC1qnmj0qXg10SxFaSj8dkL4qq3p5hXKw/WfunB51S63ac2tSuAfWGvxS8InSBxvoKqgDkpXtWQJ+zw4fzZbkV2bSzOGQHfpkjmBw9C8LaT7eLRU2nz4NbjBhY2D7Ld3zeYE2W9aysmW9Q5bvn2A7wM96UGWXiQsKZXGJ4ZBhvIObbl9izlKPp5EQpdmR70Qb/9Q0Lf9l/ERKXuRYGdQUSkgHx8RvOtPDEpUk3m6U52sYFsW7nOafJsJaafDmsfPJ5sEPUXo+c4pHfZGwuTrMuMfzXNgXC7XcwGcdQS8rlLQqmbyA2PXjvPS2Wp7O6gZMA+8TmXmjyeszY/yz0VE7ExRwmjoXXNRvH+S5rjfjm6mVXjeEU3lDjx7drcEJMrXeEPt+/Cu57/5Szl3+OpRNO4jUg4Z+0vQnuvbLnYYg+EAntdJz8rtDRr3Dt2+jDrG29lQrmKsvOgdwVVq+T4fKnlc3y5ZO6qBOF8+Co0aRPeBX8jEXfr8bc9O3H2PL6ThNNfVaVkj1H5U1pX6kpt0F1wfMKHvOy3G2qdWIl2PZvozftOxSizVPbhvBmXq3IyIgWo7wmE8HSRXWtsBSUZp5DA1rg8s5Ce1hJHK9wBh5yQw4lqqhe7gPx9O2RIpu8p4zi5FGdJGWdnxJHKT5BIpr+1G8kVO4RWRjUSeXHH4lmIZhNGGeMjaHrFf38KT+uVnOemjUuHkhLU+XcHvn3p2Hi5rcdJ9rnt8yW8Jc+8Q4RiL1zuhm9FyiU4fAnpAOxc0EkXK/FP+79NeJzlJ6d3MSh7PsRH5N51UOtkS7H6bFdTt2aLWtl5M3h4JGf1FtzVnPT69i4VnO6jcJvYfcRRKG28yxU5CslRqlFXJSXgNW1aqnRrb5X7pFHMbtPXZmpgN2VyAmbMQgW+Lc7f1pvKoq+WfWiC2CTHUgCBFNJFZ4tfcjoAnqbLL8DEZ6ox0Rs2ljGtZG2e+Sva3zNGnBt6U4QeAh9WWIl2M/ZxzT7RIHPFy6GPBvwWLpSzfuNJyJRu7zyZMDQ6zDEnHEvNlbvM3DTQVMhZcDJ5zXGq3u1j23VQ7KOXAbRSYrHTBeNNcqNrjbyw7SudmK3x9GmJWiXdzvtwa8Sk6v4C9sozWj7aACvRoOLJpIMCRq0ZNJl4cXF5fD9XITzfUwVqCOTG9Zk0MqJwxMPd7BQCmXg/CFEMs/WJSfgwjxaaljHddNqVvGi8nyNYR3cPGfRm+cb43UGTJM2/F8spXgqaEBSeD4QEB3q1l7HKg5dvB7OHN6S9n+M6l7K1qsq2x3ZKE0W5i8ed5VqJGpldQDPqs47KuY0/m8NUFVfHDI+oxSgqPCQ9GHNm3Yi/Yd5dfTdcsWzEr5bEcavlq58mkaRsQL4RxdPfUaidaPq2gC9p3B7wK6c4lk3VjGr5rPKzfVGwv6lJKrCr9BDnSaXyNkDzkUOVnXbLsBsX0k+IwNi1aoHnbRDnbPEeE4JOkSFEvWLq4mwSfqYMAgZzg9UlNRN0SdIp5u/pDoxKMuGqFn5CYNmG+xSLMrq4LE1tHr5LDVD8oLk3AYTs1X2AOdCA39ZvGLWnkJ0vHoE6EqHMahV/BQHKDyf/fnm5ShsXbSa09E1s2G+bnXArdxU5kfM4dvNpE7jR5KfQeR2VDtLbQV9vEFHJXgZIeBQlwmH2wUJvc8L5fh9FjoMl3X9KLjtgTz/Gj4uxiXiWqTQ3KI/n2K9zFhQ+h1z3gYVaSgkUn5j0IhTXWj6u8Rb2f2bSjlrbq5gIbOHUXtbWfV3KNg05TqvL+/7eQK9zDMhXk/6p9CX2W3yoQeGeN/oL+w/pq/wLr/3Xz/W/lZcBfTlMmH6r2E6XQyY0nYSvfAoIr6eq/OuBHgB/pk9Ip6tHlBTxrhOjH2H8oOFczNTrgLb0FrXt6OHzT3hvWrvKuePrPJlD8bjBC60iDdNpNXuR7ixzVn2rFGxpqEsq+5IrSLx9Ey9KxoOBOlUFk/RdVIp7tk7RBM1eNQAYlEBXUG4VOiUqTvn6y0r3yQlFDXQtWZSGgQLAB6AJQBj1gU/PwATCwe0+L5UwnJc/wtQ1OD8hfjvj2OTnlZ9Lt3kdAoKCeuBV7YcQZl4JumTr/A+jScbMAZBp+p2vdC2B14ZR7WFtBSW26x9uxIsbUSqd4J0CQBkyVWp6hkGLoPbVs2hDgTeLgGV6pJNMqp7fqXU7mOEfFrXMHuK93U8K3eTFajuDGqKnQKX6Bpo8Vxm9D1pSnv5K23Pf6NoDcsxMYmppOmqmSsRPHGNJjiS+vX4x+1EGGeGuZnVX/2LmG6sgRnmWrGmQHVUxTXpU2qIaY0K8c2lGNg+xYvPCN6lIgjMF5NpJrgnqBjjo3HZaYsvPLmAaKqPiHLX1Bh9CHpfGnWXi+7eQU8zMBOUJuPnwWlQVEkF52Sq0Cl3nkVlHnOvZkD9oYoO7KXFOI2oRj4CwKaAV7mEqbs1c8GfXM6lwtdcXT6BUfAcv8lS6rWTGlvahGQvCBdN8vC4+ezsT2cIT/rgmhmJAPGVv6rHN7jHq1X1b4NSUFzo5yoVzz2p/9UqJoQGDLBXlWXm7bn0oTBWtdEVbfLGkGvsj2BkmNPmm4+ENI+8LKm/TDxpl2q2ayQ1ndhMtEW7ciEAH12WpzCu5WWLTLB6Oc/JqoUKhtw5AXXJVTh3EImrFwYcgIqhXhgl9/6pGUaNMGuS4sCYKrM2cQn3rHhR8AnHMJ0WXqMKqWmrs1KJDNfuRkNsDX9GLCJrvcRHClTQyyMgK3tsFJLbVpPIOLQ8dx2Wr3Y4/+WWbs1rwbqFxYkcjlBXN3bBrBKe+ha5s5+cp0XZMGqrOYmg3pHQYZl2GBNwRhmprjKnA3bQnMoFHMoRyK0THVAqu3XIvA3tDWEcLvuXH6Y7VgjIVh3LJ3mQOsPRPjRvL6DjUhQCB6wEEGBJv/E+hUp6MEo7LVvEj6tzzQIzSPbQhnJEeMmhjPRw3cXvExFWWsWUmvIIH9dBdRmRMHfdaUeqqRXTu9sTRU8o9JmAzeMWTumFoV9MCQEVa7+tmTI8D4VmUoDgaOx8blzmuxa0E5Y4X8nLGgmwbLObPhTBfHw1Y64rjBJqVgpABJ+m9DgXa1Nnvj0rdk8iwV9X7sVJhJcEVVGH6/RvYMITNgF9wgRa0vtZZQ2j5hXDJ23vAKZ9EDf/uWW7NoGv9yFHV6Nnni6HQ+IYr4tzS9Xs7lU/6K2sgGGmIR3PMJ7J3xHtlb6NtZjYa1r5SnygnqbfUmZEG0FdgSkxXdwBh30AnAZW2APo7unxpX2RPtlHrZSYmhFW6pqu8VK80qLfjqlmx5oQd1IMiBBwVJ2tOP1QVWJBSpDQ+oGe1Tb4cy2pXwZCaWGgk3dwqY5kQ4XMb7LlHqsyx5FNqSpLRCtW5rhLrgL1A9ITdZXTvEA/rZahe0zEEyjz6SWwAGzgNvMJ7dPwXVFZodxN0ezpSTUvO6P1KVd0b2YDHbV8KEDkKvPFBbi+z1O3LuXtqEHEx2EhF0wy2KUQ/k5J4+1zN7Ju7F3ndWzAvJTTFV3N9lg5Q2+eXZIUp8SzTWX6KjmjDFqgZ1XvFYZM4LxJnOAe9VrgPg3fsqfcnd/CNK1C+4+Ik/gu1NT1x2r+JIcHq1VQdLhaHq2HVa9H+Rm4PN+qXAfALxtrU6nwr/AuFLE0u9eEwIv/B3OQEGvgiHBe3pAd1qqJZJsjM9uw0o5FW55fpGJL3Pn7FsArpXhjYQDDYoQNHL0SdMGprKBgXfGDeb4cHEo0k4CZS4ZZ37L4y/kg+r0yt5cRlJ0/aTpGcumzlVD2FRTYE7Zx6I8KwgkI0dU6QOJpWLUYBb0e/8mkc8OtEfH43zFCiy7w0JIMoKcLG3OqlRaVvQ9wAmGUr9jLpp7REXmXe73ZgWDnNW89d4uVYGQsk93KdZdmVaqQ+CFdaYa8yV0YkbcekRTKH5D2IDHmB0RBjDjKW/RyrNLUq2J1gLmjpHmjYNx0HQcqO/mm/CJrzdy+NqiL+n2cJLVI1Q6qSd+qQWvP4uHxqvO3W9j0tKI1E4vVAOB11E3bcP/XlAfwQMb8qN3FzM64p9WnlZNHZqPtJgHTiuLGDF5+C2tZw/SMrrNZcThmpTYBOrMCAG79AzzbeKb3+OCXAB4OfOKRrBCPh96nATrzN5D2vLDLQ3+IhnZQZzRwcDjcafYlsfSzOaUPqvaIRd8q9+aWLgyOkBAQIqVHR0Uc6ho6V3f1Bk+eF5xjeRpLIr+UxW1aUYBtSIqa2sIxUvALQg6Zq0g2b4XBrqvklF6EGRpz6Vc4N6ptm/ww4Hs7TlHtaHibMrDnSKlVm7Rf7Vk03JXHWpoQ8+vZlqbZkYdIa1lCZhEnIaL8aD2Ilb5TCkrqTShN0PS/Y1fM2p9CzSvaqiwLcyCZB8bA0XqVFhMUtVcF3mDeoKnKlOBBz/J3z7Gd8FYufZpDfLh8pp9dZxC7hn1xK07P6FKAAAAAElFTkSuQmCC\n"
            },
            "metadata": {
```

```
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "model.save('/content/drive/MyDrive/spam.h5')\n"
      ],
      "metadata": {
        "id": "oz-kGP9u0HN_"
      },
      "execution_count": 32,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "Testing the model"
      ],
      "metadata": {
        "id": "DdZJUVY83Dx8"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "loaded_model = load_model('/content/drive/MyDrive/spam.h5')\n",
        "test_loss, test_acc = accr = loaded_model.evaluate(x_test, 
y_test)\n",
        "print('Test set\\n  Loss: {:0.3f}\\n  Accuracy: 
{:0.3f}'.format(test_loss, test_acc))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "giqKxvSy0HUP",
        "outputId": "ef86c6f8-f5af-47ca-a3fa-2f29336b9e14"
      },
      "execution_count": 34,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "35/35 [==============================] - 1s 8ms/step - loss: 
0.1899 - acc: 0.9848\n",
            "Test set\n",
            "  Loss: 0.190\n",
            "  Accuracy: 0.985\n"
          ]
        }
```

```
  ]
},
{
  "cell_type": "code",
  "source": [
    "import numpy as np"
  ],
  "metadata": {
    "id": "mvhSz3tB1aqK"
  },
  "execution_count": 35,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "y_pred_prob = loaded_model.predict(x_test)\n",
    "\n",
    "print(np.round(y_pred_prob, 3))\n",
    "y_pred = y_pred_prob > 0.5\n",
    "y_pred"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "5CXUWSaQ1cA1",
    "outputId": "72e26d54-7b26-49a2-ffb5-797e8d9880a0"
  },
  "execution_count": 36,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "35/35 [==============================] - 1s 9ms/step\n",
        "[[0.006]\n",
        " [0.   ]\n",
        " [1.   ]\n",
        " ...\n",
        " [0.   ]\n",
        " [0.   ]\n",
        " [1.   ]]\n"
      ]
    },
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "array([[False],\n",
          "       [False],\n",
          "       [ True],\n",
          "       ...,\n",
          "       [False],\n",
              "       [False],\n",
              "       [ True]])"
            ]
          },
          "metadata": {},
          "execution_count": 36
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "for i in range(5):\n",
        "    print('%s => %d (expected %d)' % (x_test[i].tolist(),
y_pred[i], y_test[i]))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "5Ps52F8X1e_z",
        "outputId": "b58c7534-4c46-4751-cae6-5c100fedb447"
      },
      "execution_count": 37,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1366, 1578, 1432, 19, 7893, 19, 19, 38, 118, 1650, 19, 738, 4, 449,
3023, 35, 1285] => 0 (expected 0)\n",
            "[1, 188, 11, 6440, 2, 7, 1, 135, 2, 28, 12, 4, 290, 7931, 1,
104, 33, 3, 22, 647, 15, 28, 4, 3607, 18, 374, 191, 224, 2137, 107, 433,
9, 74, 10, 5, 1097, 1806, 1171] => 0 (expected 0)\n",
            "[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
39, 54, 258, 144, 3, 54, 21, 3428, 3, 16, 2, 173, 53, 144, 761, 264,
7182, 208] => 1 (expected 1)\n",
            "[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 33, 3, 1528, 13, 263, 53,
79, 228, 79, 3, 31, 7, 838, 69, 10, 8, 5, 168, 2, 205, 10, 54, 3, 499,
14, 8, 46] => 0 (expected 0)\n",
            "[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 715, 29, 357,
532, 622, 15, 1107, 528, 706, 49, 435, 19, 98, 563, 496, 292, 71, 521, 2,
906, 1546, 138, 1200, 2216] => 1 (expected 1)\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from sklearn.metrics import classification_report"
      ],
      "metadata": {
        "id": "h3E5TSen1kAG"
      },
      "execution_count": 38,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "print(classification_report(y_test, y_pred))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "2yWVk2gM1mn2",
        "outputId": "4f3af0dd-56ac-45bb-8040-68465e0e7328"
      },
      "execution_count": 39,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "              precision    recall  f1-score   support\n",
            "\n",
            "           0       0.98      1.00      0.99       965\n",
            "           1       1.00      0.89      0.94       150\n",
            "\n",
            "    accuracy                           0.98      1115\n",
            "   macro avg       0.99      0.94      0.97      1115\n",
            "weighted avg       0.99      0.98      0.98      1115\n",
            "\n"
          ]
        }
      ]
    }
  ]
}
```