```
{
 "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 LIBRARIES"
   ],
   "metadata": {
    "id": "RpWrXSKwfJ_J"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
```

```
    "import matplotlib.pyplot as plt\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.preprocessing import LabelEncoder\n",
    "from keras.models import Model\n",
    "from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding\n",
    "from keras.optimizers import Adam\n",
    "from keras.preprocessing.text import Tokenizer\n",
    "from keras.preprocessing import sequence\n",
    "from keras.utils import pad_sequences\n",
    "from keras.utils import to_categorical\n",
    "from keras.callbacks import EarlyStopping"
   ],
   "metadata": {
    "id": "YPzdZ4tmfPRe"
   },
   "execution_count": 1,
   "outputs": []
  },
  {
   "cell_type": "markdown",
   "source": [
    "###READING DATASET"
   ],
   "metadata": {
    "id": "zeNL9YMSfUJK"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "from google.colab import drive\n",
```

```
   "drive.mount('/content/drive')"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "HauemKH5fWPv",
   "outputId": "c83100ba-baca-4728-be11-b9bc529793b6"
  },
  "execution_count": 2,
  "outputs": [
   {
    "output_type": "stream",
    "name": "stdout",
    "text": [
     "Mounted at /content/drive\n"
    ]
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "df = pd.read_csv('/content/drive/MyDrive/spam.csv', delimiter=',',encoding='latin-1') \n",
   "df.head()"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 206
   },
```

```
  "id": "qHDVTMuQfa70",
  "outputId": "29b69d7e-d09a-40ab-873f-d5ad243e9671"
},
"execution_count": 4,
"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",
       "  Unnamed: 3 Unnamed: 4  \n",
       "0       NaN       NaN  \n",
       "1       NaN       NaN  \n",
       "2       NaN       NaN  \n",
       "3       NaN       NaN  \n",
       "4       NaN       NaN  "
     ],
     "text/html": [
       "\n",
       "  <div id=\"df-aacb55b5-5ff8-4df6-b665-2f971d44cefc\">\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",
```

```
"  </tbody>\n",
"</table>\n",
"</div>\n",
"    <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-aacb55b5-5ff8-4df6-b665-2f971d44cefc')\"\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-aacb55b5-5ff8-4df6-b665-2f971d44cefc 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-aacb55b5-5ff8-4df6-b665-
2f971d44cefc');\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": 4
    }
   ]
  },
  {
```

```
 "cell_type": "markdown",
 "source": [
  "###PRE-PROCESSING THE DATA"
 ],
 "metadata": {
  "id": "MkonaHOOgnj7"
 }
},
{
 "cell_type": "code",
 "source": [
  "df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)\n",
  "from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator\n",
  "X = df.v2\n",
  "Y = df.v1\n",
  "le = LabelEncoder()\n",
  "Y = le.fit_transform(Y)\n",
  "Y = Y.reshape(-1,1)\n",
  "X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)\n",
  "max_words = 1000\n",
  "max_len = 150\n",
  "tok = Tokenizer(num_words=max_words)\n",
  "tok.fit_on_texts(X_train)\n",
  "sequences = tok.texts_to_sequences(X_train)\n",
  "sequences_matrix = pad_sequences(sequences,maxlen=max_len)"
 ],
 "metadata": {
  "id": "JeztGhrbgpZw"
 },
 "execution_count": 5,
 "outputs": []
```

```
},
{
 "cell_type": "markdown",
 "source": [
  "###CREATING MODEL"
 ],
 "metadata": {
  "id": "M2b-s_cCgvRU"
 }
},
{
 "cell_type": "code",
 "source": [
  "inputs = Input(shape=[max_len])\n",
  "layer = Embedding(max_words,50,input_length=max_len)(inputs)"
 ],
 "metadata": {
  "id": "_UVvwrn5gwmw"
 },
 "execution_count": 6,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "###ADDING LAYERS"
 ],
 "metadata": {
  "id": "skV8YioigzgQ"
 }
},
```

```
{
 "cell_type": "code",
 "source": [
  "layer = LSTM(128)(layer)\n",
  "layer = Dense(128)(layer)\n",
  "layer = Activation('relu')(layer)\n",
  "layer = Dropout(0.5)(layer)\n",
  "layer = Dense(1.5)(layer)\n",
  "layer = Activation('sigmoid')(layer)\n",
  "model = Model(inputs=inputs,outputs=layer)"
 ],
 "metadata": {
  "id": "IuZGN5uzg2IN"
 },
 "execution_count": 8,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "model.summary()"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "ezhwBEQYg5GT",
  "outputId": "5435542d-4e51-40ea-96c5-cfbda7e5bbd9"
 },
 "execution_count": 9,
 "outputs": [
```

```
{
 "output_type": "stream",
 "name": "stdout",
 "text": [
  "Model: \"model\"\n",
  "_________________________________________________________________\n",
  " Layer (type)                Output Shape              Param #   \n",
  "=================================================================\n",
  " input_1 (InputLayer)        [(None, 150)]             0         \n",
  "                                                                 \n",
  " embedding (Embedding)       (None, 150, 50)           50000     \n",
  "                                                                 \n",
  " lstm (LSTM)                 (None, 128)               91648     \n",
  "                                                                 \n",
  " dense (Dense)               (None, 128)               16512     \n",
  "                                                                 \n",
  " activation (Activation)     (None, 128)               0         \n",
  "                                                                 \n",
  " dropout (Dropout)           (None, 128)               0         \n",
  "                                                                 \n",
  " dense_1 (Dense)             (None, 1)                 129       \n",
  "                                                                 \n",
  " activation_1 (Activation)   (None, 1)                 0         \n",
  "                                                                 \n",
  "=================================================================\n",
  "Total params: 158,289\n",
  "Trainable params: 158,289\n",
  "Non-trainable params: 0\n",
  "_________________________________________________________________\n"
 ]
}
```

```
 ]
},
{
 "cell_type": "markdown",
 "source": [
  "###COMPILE THE MODEL"
 ],
 "metadata": {
  "id": "EL-E1UEHhAfZ"
 }
},
{
 "cell_type": "code",
 "source": [
  "model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['accuracy'])"
 ],
 "metadata": {
  "id": "crQvkygFhCOj"
 },
 "execution_count": 10,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "###FIT THE MODEL"
 ],
 "metadata": {
  "id": "yP-NdpLChGtB"
 }
},
```

```
{
 "cell_type": "code",
 "source": [
  "history = model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,validation_split=0.2)"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "7rHd1ru-hJOM",
  "outputId": "1a8279e9-9977-427b-f2ec-4978060ff303"
 },
 "execution_count": 11,
 "outputs": [
  {
   "output_type": "stream",
   "name": "stdout",
   "text": [
    "Epoch 1/15\n",
    "168/168 [==============================] - 35s 182ms/step - loss: 0.1809 - accuracy: 0.9414 - val_loss: 0.0834 - val_accuracy: 0.9713\n",
    "Epoch 2/15\n",
    "168/168 [==============================] - 32s 189ms/step - loss: 0.0383 - accuracy: 0.9880 - val_loss: 0.0865 - val_accuracy: 0.9749\n",
    "Epoch 3/15\n",
    "168/168 [==============================] - 30s 179ms/step - loss: 0.0183 - accuracy: 0.9955 - val_loss: 0.0943 - val_accuracy: 0.9725\n",
    "Epoch 4/15\n",
    "168/168 [==============================] - 33s 196ms/step - loss: 0.0097 - accuracy: 0.9973 - val_loss: 0.1148 - val_accuracy: 0.9761\n",
    "Epoch 5/15\n",
    "168/168 [==============================] - 32s 192ms/step - loss: 0.0058 - accuracy: 0.9985 - val_loss: 0.1210 - val_accuracy: 0.9713\n",
```

```
        "Epoch 6/15\n",

        "168/168 [==============================] - 34s 202ms/step - loss: 0.0058 - accuracy: 0.9988 - val_loss: 0.1246 - val_accuracy: 0.9773\n",

        "Epoch 7/15\n",

        "168/168 [==============================] - 32s 190ms/step - loss: 0.0041 - accuracy: 0.9994 - val_loss: 0.1380 - val_accuracy: 0.9701\n",

        "Epoch 8/15\n",

        "168/168 [==============================] - 30s 179ms/step - loss: 0.0070 - accuracy: 0.9982 - val_loss: 0.1345 - val_accuracy: 0.9713\n",

        "Epoch 9/15\n",

        "168/168 [==============================] - 32s 189ms/step - loss: 0.0121 - accuracy: 0.9961 - val_loss: 0.1338 - val_accuracy: 0.9737\n",

        "Epoch 10/15\n",

        "168/168 [==============================] - 31s 184ms/step - loss: 0.0034 - accuracy: 0.9994 - val_loss: 0.1521 - val_accuracy: 0.9749\n",

        "Epoch 11/15\n",

        "168/168 [==============================] - 32s 189ms/step - loss: 0.0026 - accuracy: 0.9991 - val_loss: 0.2803 - val_accuracy: 0.9390\n",

        "Epoch 12/15\n",

        "168/168 [==============================] - 32s 193ms/step - loss: 0.0056 - accuracy: 0.9982 - val_loss: 0.1374 - val_accuracy: 0.9737\n",

        "Epoch 13/15\n",

        "168/168 [==============================] - 30s 180ms/step - loss: 0.0025 - accuracy: 0.9994 - val_loss: 0.1648 - val_accuracy: 0.9737\n",

        "Epoch 14/15\n",

        "168/168 [==============================] - 32s 192ms/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.1707 - val_accuracy: 0.9737\n",

        "Epoch 15/15\n",

        "168/168 [==============================] - 33s 194ms/step - loss: 0.0017 - accuracy: 0.9997 - val_loss: 0.1756 - val_accuracy: 0.9737\n"

       ]

      }

     ]

    },
```

```
{
 "cell_type": "code",
 "source": [
  "metrics = pd.DataFrame(history.history)\n",
  "metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accuracy'}, inplace = True)\n",
  "def plot_graphs1(var1, var2, string):\n",
  "  metrics[[var1, var2]].plot()\n",
  "  plt.title('Training and Validation ' + string)\n",
  "  plt.xlabel ('Number of epochs')\n",
  "  plt.ylabel(string)\n",
  "  plt.legend([var1, var2])"
 ],
 "metadata": {
  "id": "A5EOG8oNhMkV"
 },
 "execution_count": 12,
 "outputs": []
},
{
 "cell_type": "code",
 "source": [
  "plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 295
  },
  "id": "kAGDb_G1kB2C",
  "outputId": "ea39a4cd-b96c-4ec7-d8df-6cc6af26f6b5"
```

},
"execution_count": 14,
"outputs": [
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYgAAAEWCAYAAAB8LwAVAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA
6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAgAElEQVR4nOzdeXiU5dX48e9JQgIhYUkCCRCWgMgaFomiog
Vcse67uKK2Lq/W1tb6c3lftVprF7totYu14lpRUREErbqDWrbWASmQVlACBJGRhCUnIen5/3M+EIWSZhEx
mJjmf68o1M882ZybJc557ee5bVBVjjDGmoahQB2CMMSY8WYIwxjhTKEsQxhhhjGmUJwhUhhjTKMsQRhj
GmUJQhjjDGNsgRhAiYib4rIFe29bSiJSI6InBCE434434gIt/znl8iIu8Esm0m0b3meIiOwRkei2xmpMUyxBdHLeycP
3UyciFX6vL2nNsVT1T1FFV9qr23DUCcicpuIfNFN8YEeS1WfU9qBGDDLjf
jzxJEJ+edPBJUNQHYDJzut+w533YiEhO6KMPSs8DRIpLRYPlFwFequanjIEMXUZ9vcYHixBdFEiMkNEckXk/4l
IPjBXRPqKyD9FpFBEdnjP0/328a82mSMiyQcYfpFHReTZJuIOJMb7ROQT7
3jviEiK3/rLRGSTiBSLyJ1NfT+qmgu8ChF8BZ2lGYqqTJHHfRFeTzJuIOJMb7ROQT7
3jviEiK3/rLRGSTiBSLyJ1NfT+qmgu8B1zWYNXlwNMtxdEg5jki8rHf6xNFZK2I7BKRRwDxWzdcRN4z4isSk
edEpI+37hgCPC6VwK8VUSGiYj6TqgiMlBEFopIiYhsEJHv+x37HhF+x37HhF5UUSe9r6r6bSSKS1dR3ICIPicgWEdktIst
F5Fi/ddEicoeIfOMdDm7m97YfK9/eI3lRIhR7S0JmRTYZ9WGDTu7eEYCkTkDm/5kyLyc79jzBCRXL/XOd7fYzZQJiIxXknO9x6rReTsBjF+
X0TW+K0/TER+KiIvN9juRYF5qKnPahpnCaJrSwOKgKHHANbi/h7aOhBxl4h/h7ne6yFABfI/tPBdY
KNwZX8nm5FHAZAVe2ni/wAOcAJ3vMZQBXQvZntJwd/F5/F1aocJvGi
4FnA/xMjcX4v36v/wd4y3t+FzDPb11P7zs4o1/cDr7Xxu/rYe3458B+/7QR3Qv9eE8c9C/iis
d+h93qY913G4JJJLZDot/4B4Env+T3Ayr91Y4GKVvz97Ames/XAWc2ss1s/3gbrHsS+Lnf6xlAboPPPdlUL
MXzpe1/gbeCHTWz3JvB97/lpwOqO+B/rbD9WguJaClV1r++rkE7yq0Dd+VT8MY1P+1iKSKyDwDwR2
eod91lcSSMQvu+y1G/ZJmCQ3+uG3013aaK+X0Ru8apvdonITI7FEdvsE4wR+5cqyYrrIE7Ku5BL
u9GMYHEAO40t+l3vNLgWcOIqYuyxJE19ZwKN+fAKOAqaraC/iOt7ypaqP2kAckiUi8iXwpIj
/be8/kFvZ5CrgAOBFIBF4/yDgaxiDs/3l/gfu9ZHrHvbTBMZNSbnHqHx4/NobbsV99Fs2BNjaQkwH8NobbsV99r6q2g
fY5RfLFmBEI7tuAYY3cdgyXKnOD0D0MMSsbfnkb7rtM9Fs2BNjaQkwH8NobbsV99r6q2g
fY5RfLFmBEI7tuAYY3cdgyXKnMJ62Rbeo/n4gMBf4G3AgkezGsDCAGgAXABHG9zU4DnmtiO9MMSxDG
XyKuLn2niCQBdwf7DD4E7DVV1E7AMuEdEYkXKkXKOD0IMU4HzhRRI4RkDjVCbDGIuFbRsnK4q6c4q8QiBGRu3D1/D6PA/eKyEE
d6V+03sf6JMBPYAu0RkEkPDTBvsX0mX0ViKu6q8Qi
BGRu3D1/D6PA/eJyEhxJohIMvBPYICI/EhE4kQkUUSO8kTpRBWQmevt8CXxxRJJEJA34Lwj+EWEZnixXCl1TwSsb
8dq3VHVzG76DLs8ShPH3B6AHUAR8AjJt8NZsR2ioCArgBt+wJ408XJ16bgv7KK5a
aSj7N3K2KQ5VLQLOB36J+7wgjU/8NvkZcbBjuav0NXIO2vweA//WqXG5p5C1m49oltgGvAner6uJAYmvg
bdxn+hpXTbWX/at/FMHTbWX/at/fge8CLyDa6f5O9DDq946EDq9Z9Ez35mdSrjP9/
70pVX8K1C/0DKMWVGpL8DvGUt49VL7WReI04xoQNEXkBWKuqQS/BS3+9nIxZ05o6ZBHmqjMqrxOFE
3Iicri4/v9RIjILOBN3NWhMm4hhFK4r7jxLDm1dyuacJCGq0pJxlX5XK+qXqkwqQzRBOpRKQnrkpqEzArxOFE

NKtiMsYY0yirYjLGGNOoTlPFlJKSosOGDQt1GMYYE1GWL19epKr9GlvXaRLEsGHDWLZsWajDMMaYiCIi
m5paZ1VMxhhjGmUJwhhjTKMsQRhjjGmUJQhjjDGNClqCEJEnRGS7iDQ6NaM3uNbD4ma9yhaRw/zW
XSEi672fsJ/43hhjOqNgliCepPm7GE/BDVQ2Ejeb2Z8B/EbGnAocAdwtIn2DGKcxxphGBC1BqOqHQEkzm
5wJPK3Of3CTrQwATgbeVdUSVd0BvIvdLm+MMR0ulPdBDGL/4YNzvWVNLT+AiFyDK30wZMiQ4ERpjDF
NUFWqauuoqKqlrKqWiqoayiprKa+qpbyqpsFjLZXVtUGJI613Dy6e2v7nwIi+UU5VH8NN5EJWVpYNKmU
CVlFVS9GeSgr3VFJUWknRniqK9lQSHSWMTktk9IBeDOzdHTfhW9fxxeYdvLd2O7HRUcTHxRAfG+39xNA
zNpoe3nPf8p5xMcTFRHWK76mqpo5vCvewNn83G7bvoXSvO9lXVHsn+Mpayqtr3GNVLWVVNVRU1VJ
T17pTTzC+qkmD+3S6BLGV/adaTPeWbcVNZu6//IMOi8pEJFWlrKrWO9lXeif/qv1e+5JAUWklZVUtX8kld
o9hTFovRg9lZFRaIqPTejEqLZGEuIi+rmrU8k0l/GHxej5aX9TqfaME4mNj6BEb7SUR/8QSTU9vXXLPWAYn
xTMkKZ6hyT3pnxhHVFTHJxZVZXtpJWvydrM2v5S13uOG7XvqT/YxUUJC95j62H3JMTWxOz2S932mnn
H7J8x9z2O8BOp9H92iiY+LJjY6spJpKP/SFwI3isg8XIP0LlXNE5G3gV/4NUyfBNweqiBN8NXW6X7F8LLKGi
qqvUe/onu53/M9lTUUllb5nfwr2Vtd1+jx+8Z3IyUhjpSEOCak9yElIZaUhDj6JcSRkhhbvy45IZaqmjq+LihlT
V4pa/N3sy6/lFc/30ppZU398YYkxdeXMsZ4j0OS4okOwcnuYH32bTEPv7eeTzYUk9wzlttOGc2lRw4lNjqK
iip3xVxWWev9Htzvw//quf7R70rb97x0bw3bd1fWb7OjvAr/i+24mKj6hOH/MzQ5nsFJ8XTvFn3Qn6+iqp
avC9zv0vc7XZtfys7y6vptBvbuzugBvThudP/63+mwlJ50i7a7AIKWIETkeVxJIEVEcnE9k7oBqOpfgEXAd4E
NQDlwpbeuRETuA5Z6h7pXVZtr7DZhZld5df0/4vrtpeyqqKG88sD6WN/zyprGT+xN8VVtJPeMpV9iHBkp
PetP+ikJcaQkxtW/TuoZ26p/9LiYaKYMTWLK0H0zV6oqW3dWsNY7wazxrjoXrymoP+F17xbFqFRXyhg9
wHtMS6Rvz9hWfbaOoKr8+5tiHlqyns82lpCSEMf/njqGi6cOIT523ykhNiaK3u5ftl1U19axbWcFm4rL2VxS
zpaS8vrn/91Ywh6/JAyQ2iuOIUkuWQxN6smQ5B4MSerJkKR4UhJi97sSr6tTcndUsCZ/N2vzSllX4B43Fpf
hm9EgPjaaUWmJnDJ+gEvwXqmwd3z7fcbOptPMB5GVlaU2WF/Hqq6tY2NRGWvy3JW2r7i+bdfe+m16
9+hGUs/YJovg8XHRxHeL8YriTRTT/Yrz3WOiQ1It0Zi91bWsL9jDmnzf53dXqSVlVfXbpPXqzqi0RKYOT+Kk
sWkc0j8hZPGqKh9vKOLhJetZmrOD/olxXDd9BBdPHdIuV+sHG9uO8mo2FZcdkDw2l5STv3sv/qeq+Nho
hiTFk963ByVlVazLL62vNhSBYck9XcL2kvWYAYkM7hsfNn874URElqtqVqPrLEGYQBSWVrpSQV5p/VXah
u17qKp1V/8xUcIh/RPqq15GpyUyZkAv+ifGRVSd68FSVQr3VNaXNtbmlbLaq+MGGN6vJyeNTePkcalMT
O/TIScsVeWDrwt5eMl6vti8kwG9u3P9jBFckDU45IkhUHura9m6s4LNXtLwJY/cHeX07tGNMd7f3OgBvT
g0NWG/kpBpniUIE7C91bVs2L5nXwOed3VctGffVXFqr7j6qhRfl+7wlARiY6zOtil5uyp4d3UB76wq4D/fFl
NTp/RPjOPEsamcPC6NI4cnt/v3p6osWbOdh99bT3buLgb16cH/zBzBeVPSiYuJjMRggs8ShGlSwe69fLS+i
E83FJG9dRcbi8qo9SrWffXqvh48vuJ6UhjWq0eSXeXVvLfOJYsP1hVSUV1LYlwMM0f356RxqcwY1f+gekr
V1SnvrC7gj++tZ9W23QxO6sENMw7hnMPSLYmbA1iCMPXKKmv4bGMxH60v4uP1RazfvgeA5J6xTB7Sl
7ED9lURDU3uGZE9cyLJ3upaPtlQxNur8lm8ZjslZVXERkcx7ZBkThqXxgljUumXGBfQserqlLdW5fPwkvWs
zS9lWHI8N8w8hLMmD7IeOaZJliC6sJraOrK37uJjLyF8vnkHNXVKXEwUR2QkcezIFKYdksKYtF7WgBditXX
K8k07eGdVPm+vzmdLSQUiMGVIX04al8pJY9MYltKz0f3e+CqPPy5Zz/rtexjeryc/OO4QTp8wkBhLDKYFli
C6EFUlp7icj9cX8tH6Iv79bTGle2sQgXEDe3HMIf04dmQKU4b2jZgGyq5IVVmbX8o7qwp4e1U+q/N2AzA
qNbE+WYwZkMjr2dv443sb+LawjEP6J/CD4w7htAkDreRnAmYJopMrKavikw2uhPDxhiK27qwAYFCfHh
w7MoVjRqZw9lgUazuIYFtKynl3tUsWS3NKqFPXRrS3uo5RqYncdPxIThmfZqVA02qWIDqZvdW1LMvZwc
cbivh4QyGrtu1G1Q0NcfSlZI4Z2Y9jD0lhaHJ8l+pi2lWUlFWxZE0BS3NKOG50f04aa4nBtJ0liE7kkffW88f3
NlBZU0dMlHDY0L4cc4grJUwY1NvqnI0xrdJcgrC7SSLIM//O4cF3vuaksalcdMRgpmYk07MTDhxnjAkPdn
aJEG+tzOOuhas4YUwqf7rkMCspGGOCzs4yEWBpTgk3zfuSSYP78MfZky05GGM6hJ1pwtz6glKufnIp6X1
68PcrDqdHrHVNNcZ0DEsQYSx/116ueOK/xHWL5qmrjrBuqsaYDmUJIkztqqhmztz/sntvDXPnHM7gpPh
Qh2SM6WIsQYShypparn1mGRu27+Evl05h/KDeoQ7JGNMFWS+mMFNXp/z4xRX859sS/nDhJI4ZmRLq
klwxXZSVIMKIqvLzN9bwRnYet58ymrMmDwp1SMaYLswSRBh5/KONPPHJRq6cNoxrvjM81OEYY7o4Sx
Bh4rUvt3L/ojWcmjmA/zt1rI2hZIwJOUsQYeCTDUXc8tIKpmYk8dsLJtrAa8aYsGAJIsRWbdvFtc8sZ3hKAo
9dnmVzNBhjwoYliBDaUlLOnLlLSewew5NXHU7vHt1CHZIxxtSzBBEiO8qquGLuf6msruWpq45gQO8eoQ
7JGGP2E9QEISKzRGSdiGwQkdsaWT9URJaISLaIfCAi6X7rfiUiK72fC4MZZ0erqKrl6qeWkrujgsevOJxDUxN
DHZIxxhwgaAlCRKKBR4FTgLHAbBEZ22CzB4GnVXUCcC/wgLfvqcBhwCRgKnCLiPQKVqwdqaa2jh88/wVf
bNnJQxdO4oiMpFCHZIwxjQpmCeIIYIOqfquqVcA84MwG24wF3vOev++3fizwoarWqGoZkA3MCmKsHU
JVuWvhKhavKeCe08dxSuaAUIdkjDFNCmaCGARs8Xud6y3ztwI4x3t+NpAoIsne8lkiEi8iKcBMYHDDNxCR
a0RkmYgsKywsbPcP0N4eeW8D//hsM9fPGMEVRw8LdTjGGNOsUDdS3wJMF5EvgOnAVqBWVd8BFgGf

As8D/wZqG+6sqo+papaqZvXr168Dw269F5du4bfvfs05kwdx68mjQh2OMca0KJgJYiv7X/Wne8vqqeo2V
T1HVScDd3rLdnqP96vqJFU9ERDg6yDGGlTvr93O7a9+xbEjU/jVeRPsLmljTEQIZoJYCowUkQwRiQUuAhb
6byAiKSLii+F24AlvebRX1YSITAAmAO8EMdag+XLLTv7nuc8ZMyCRP186hW42XagxJkIEbbhvVa0RkRuBt
4Fo4AlVXSUi9wLLVHUhMAN4QEQU+BC4wdu9G/CRd6W9G7hUVWuCFWuwbCwq46onl5KSGMsTcw
4nIc5GVzfGRA5R1VDH0C6ysrJ02bJloQ6j3o6yKs589BP2VNYw/7qjGN4vIdQhGWPMAURkuapmNbbOL
mmD5O1V+WwuKWfeNUdacjDGRCSrEA+SjcVlxEZHcfgwuxHOGBOZLEEESU5RGYOTehBtQ3cbYyKUJY
gg2VRcTkZKz1CHYYwxbWYJIgjq6pSc4jKGJluCMMZELksQQVBQupe91XUMsxKEMSaCWYIIgpyicgAyrAR
hjIlgliCCIKe4DIChyfEhjsQYY9rOEkQQ5BS5Lq4D+9gsccaYyGUJIghyissYkhxvXVyNMRHNEkQQ5BSVM8y
ql4wxEc4SRDurq1M2lZQxzBqojTERzhJEO/N1cR1qXVyNMRHOEkQ721jkejBZF1djTKSzBNHONhW7eyC
GpVgbhDEmslmCaGe+Lq4DelsXV2NMZLME0c6si6sxprOwBNHOXBdXa38wxkQ+SxDtyDeKq90DYYzpD
CxBtKOC0r1U1tgorsaYzsESRDvydXG1KiZjTGdgCaId+Yb5ti6uxpjOwBJEO9pUXEZsTBQDrYurMaYTCGqC
EJFZIrJORDaIyG2NrB8qIktEJFtEPhCRdL91vxaRVSKyRkQeFpGw7ze6saiMIUnxRFkXV2NMJxC0BCEi0cCj
wCnAWGC2iIxtsNmDwNOqOgG4F3jA2/doYBowARgPHA5MD1as7WVTsXVxNcZ0HsEsQRwBbFDVb1W
1CpgHnNlgm7HAe97z9/3WK9AdiAXigG5AQRBjPWjWxdUY09kEM0EMArb4vc71lvlbAZzjPT8bSBSRZFX
9Ny5h5Hk/b6vqmoZvICLXiMgyEVlWWFjY7h+gNfJ3WxdXY0znEupG6luA6SLyBa4KaStQKyKHAGOAdFx
SOU5Ejm24s6o+pqpZqprVr1+/joz7AL55qDMsQRhjOomYIB57KzDY73W6t6yeqm7DK0GISAJwrqruFJH
vA/9R1T3eujeBo4CPghjvQfF1cR1qVUzGmE4imCWIpcBIEckQkVjgImCh/wYikiIivhhuB57wnm/GlSxiRK
QbrnRxQBVTOMmxLq7GmE4maAlCVWuAG4G3cSf3F1V1lYjcKyJneJvNANaJyNdAKnC/t3w+8A3wFa6d
YoWqvh6sWNtDTlEZQ62LqzGmEwlmFROqughY1GDZXX7P5+OSQcP9aoFrgxlbe8spLmOodXE1xnQioW
6k7hTq6pRNxeVk2BAbxphOxBJEO/B1cbUShDGmM7EE0Q5yiqyLa4fasx1WzIO62lBHYkynFtQ2iK4ip9g
3iqsliKAr2gDPngM7N8Hmf8Npf4DwH6bLmIhkJYh24OviOqBX91CH0rnlLoMnToKqPTDpElj+JCy+J9RRG
dNpWQmiHWy0Lq7B9/U78NIV0LMfXPYqJA2HmO7wyR+gRx845uZQR2hMp2MJoh1sKi6L7OqlqjJ3NZ
40AkbNCnU0B/riWVh4E6SNh0vmQ0J/t/y7D8LeXa4U0aMvTJkTyiiN6XQsQRwkXxfX6YeGdiyoNqmthi+
egQ9+CXu8wXKnzIGTfwGxYZDwVOHDB+H9n8PwmXDhMxCXuG99VBSc/Reo3A2v/wjiesH4c5o+njGm
VawN4iBF5CiuqrBqAfzpSPjnzdA3A+a84applj8Ff50OeStCG2NdLbzxE5ccJlwIF7+4f3Lwie4G5z8FQ46EV
66BDYs7PlZjOqmAEoSIvCIip/qNm2Q8vi6uETNR0MYP4fHjXX1+VAzMngdXvQXDjoET7oErFroqp78dD5
/+EerqOj7G6goX37K/w7Qfwll/gZjYprePjXefo/9oeOEy2PxZx8VqTCcW6An/T8DFwHoR+aWIjApiTBFloz
fM934lCFVY9xYs/AGsfBmqykMUnZ/8r+DZc+Gp06E0H878E1z/KYw6Zf9uohnfges/gUNPhnf+13UpLc3v
uDgrdsAzZ8Oaf8KsX8KJ97qqpJb06AOXvgKJA+Af50P+yuDH2llU73Ulxi//AW/fCW/eZveYGCDANghVXQ
wsFpHewGzv+Rbgb8CzqlodxBjD2qbi8v27uG7+DBbf7froR8fB509DbAKMOQMmXOBOwFHRHRfgjhx47
3746iXo3htOvA+O+D50a2bU2fgkuPBZ13D91u3w56PhzEddMgmmXbkuiZV8C+c90fr2hIT+cPkCeGKWS
zJXvQXJI4ITayRSdd9xwSrYvso9FqyCovWgXkKQKNA6OOxySG04Q7DpagJupBaRZOBS4DLgC+A54BjgCt
yorF1SfRfXonWw5F5Y9wYkpMFpv3d99bd8BtkvwurXYMU/3LrM81yySJsQvJu8yorgw9/A0r+7qqRjfgTT
fuSutAMhAllXwtBp8PJV8PxFcPj34KSfN59c2qpgtUsOVXvg0pddIm2LPkPgsgUwdxY8cxZc9Tb0Gti+sUaC
yj2wfQ0UrNyXCApWQeWufdv0GQqp42DM6e4xdTzUVMJfprn9LEF0eaKqLW8k8iowCngGeFJV8/zWLV
PVrOCFGJisrCxdtmxZh7/v7Adf5gZ5kWP2vONKCtN+CEdef2AvoOq98PVb7kr+67ehrhr6jYbM891P36Ht
E1DlHvjPn+CTh6G6DCZfBjNuO7iTZE2lS37/fsTFfO7fXZfT9pLzCcybDTE9XHJoj2Nv+wKePB16D4Ir33Slos
6ortaVEvdLBCvdMp/YRC8BjNuXCPqPge69DjxebQ38YiBMvcZdDJhOT0SWN3UODzRBzFTV99s9snbU4
QmivAT96PdUfvpnYqIg5shr4difBHYiKi+B1QtcyWLzv92yIUfDhPNh7FltO5nVVMHnT8G/fg1l291V4XF3
Qb9DW3+spmxYAguuh4qdcOLPYOp1B18CWv0avPx978r/FffYXjZ+5EolaePh8tca7wUViXbnwcr5sHqhS
wbVXhuXREHyIfsSQX/vsc+Q1v2eHpvhqiMvfy0o4ber138U+h534aD/WDjr0Tbt2h4J4gbgOVXd6b3uC8x
W1T+1KaIg6LAEUV0Bn/0VPv4dunc3L9ceS8zxd3DWjKPadrwdm1ypIvsFKPoaorq5BuIJF8DIk6FbC8N31
NXB6ldhyX2wY6OrEjrhZzD48LbF05KyInjtRvj6TTjkBDjrz/tuXGut//4NFv0U0g+Hi18IzlX+2kXwwqUwbBp
c/FLL32e42rsb1rzu/k42fggoDJwMQ47alxD6jW6f6r/XboR1b8JPN4T3OFcVO+BXw6DfGOidHupoQitlJM
x6oE27tkeC+FJVJzVY9oWqTm5TREEQ9ARRWwNfPuduKivdBiNP5stDf8hZL+/kH9+bytGHpBzc8VXdlVD
2i+7qcE8BxPWGsWe4+wCGTjuwN88377sG8bwV7mrxhHtg5InB/6dWhaWPu15OcYmuR9ShJ7Vu//fug
49+C6O+66qsYoM4l8aKF+DVa2D0ae6eiegIuT+0pgq+WeL+JtYtgpq90HeY+3vIvABSDgnO+372GLz5U/j
xWug1IDjv0R42fuh65V36srtYMW3SXIII9D8lWkREvWwiItFAMx3TOxFVWPsGLPmZu8JPPxzOfRyGTWP

VZ5uAnQxtj5vkRGDgJPdz0n2w8V/uxLDqVXe3c69BXuP2he4O6MX3wLfvQ+/B7j6BCRd0XO8oEdcTatgx
MP9q16106nWu5NLSFXptNbz+Q5dsp8yB7/42+CfsiRfC3p3w5q3w+k1wxiOBdZ0NBVXIXepKCitfgYoS6J
Hk2pImXOD+/oJ9AeBrA8r/KrwTRF62e0ybGNo4OrFA/zPfAl4Qkb96r6/1lnVumz6Fd++G3P9CyqFw4XM
w+tT6f9CcojLigjGKa1Q0jDjO/Zz6O3f1mP0ifPoIfPKQ26ZHkhsSI+vq0FWb9B8D33/PlWI++4ur8z/38aZ7v
1TugZfmwIZ3YcYdMP3WjqvCmHqtazv54BfQvQ+cfH94VZ8UrXe/469edA3MMd3d31rmBXDI8e6O8Y6
SOs49FnzVupJhR8vPhsSBkBCBw9xEiEATxP/DJYXrvdfvAo8HJaJwULAKFv8M1r/tbrw6/WHXZbXBlW5Oc
TlDk4M8imtsvCs5ZJ7n6v9XverudM660jUkhlq37nDKr1wRf8H18LeZrvfL4d/b/wRcVgTPnQ95X8LpD4V
mYL3pt7p66/886gb3m/7Tjo/B357t7kbK7Bdh2+eukTnjOzD9/7nqsMZ6GXWE7r1dVVb+V6F5/0DlZcOA
CaGOolML9D00pRMAACAASURBVEa5OuDP3k/ntXMzvP8ArHjeDfx2wj1wxLVN1o/nFHXwKK49U1zVTj
gaeaK7M3vB/8CiW9yYSGc+6mIu2ejuyN6dBxf9I/g33DVFxJW69u50Yzz16NPx32dVmauyzH7BtSFprbsf
5qT7Yfy54VOlkzo+vBNEVTkUrXO99UzQBJQgRGQk8AAwFqivz1DV4UGKq2OVl7gG0/8+Bggc/QM3cF0z
vWrq6pRNJeXMHN3GHjydUUJ/uOQl18vr3f+DPx3lrtr/9Suoq3HjPA0+IrQxRkW5Noi9u1wi697b1e0HU
20NfPuBSwpr/+m6pfYe4m5ezLzAjSEVbtImuERWVRYeI/s2tH21u+PbShBBFWgV01zgbuD3wEzgSjrLSL
DF37h+31V7YNLFMOP2gLrM5e3eS1VNXeQM0tdRRODI61wD9stXu5Nw7yGup0l73pNxMKJj4Ly58Nx5
8Op1rrTYXvNg7N3l7gr3v3Ft+2r399W9j+tkMOECGHxk+DaUA6RlAuo+S7C6TB8M370PA6yBOpgCTRA9
VHWJ15NpE3CPiCwH7gpibB0jabirz5842zW6BmjfKK5B7J4ZydLGwzUfuAHgRp8KiWmhjmh/3brD7Ofh
qTPcyLGXvuLulQhUbQ2UfOMlgtX7ksGuzfu26d7HVdVMuti1LYw8CWLi2v+zBIOvJ1PBV+GbILr3cb34TN
AEmiAqvaG+14vIjcBWIKGlnURkFvAQEA08rqq/bLB+KPAE0A8oAS5V1VwRmYkrrfiMBi5S1QUBxhs4ETd
iaCvlNDaKq9lftx5w+NWhjqJpcYluhrq5p7ixpq543XUzbmhP4f6D2xWshO1robbSrY+Kcb3cBh/hLjZSx7u
eQL0GhldPqdboPdhVv4VrO0S+10Adqd9vhAg0QfwQiAduAu7DVTNd0dwO3r0SjwInArnAUhFZqKqr/TZ
7EHhaVZ8SkeNw7RyXecN6TPKOkwRsAN4J+FN1AF8X17T27uJqOlbPZDfH9ROz3LAc5/zVJQT/KqKy7fu2
T0h1J/+p1+xLBCmHRk7JIFAikJoZngmittqV2sK1w0Yn0mKC8E70F6rqLcAeXPtDII4ANqjqt95x5gFnAv4JY
izwY+/5+0BjJYTzgDdVNQwmVdhnY1EHdHE1HaP3IG+Y8JNdkgB3H0K/0a5ayH+gu54Hecd8JEnLdMPV1
9V27BD1LSn62pXeBjRS2jPtqsUEoaq1InJMG449CNji9zoXmNpgmxXAObhqqLOBRBFJVtViv20uAn7X2B
uIyDXANQBDhrTjIG8B2FRcRoZVL3UeySNcm8nW5W5sn6ThkTMkR7CkZboRgUs2Bm9Yj7aob6C2HkzB
Fuh/wBcishB4CSjzLVTVVw7y/W8BHhGROcCHuLaN+qmsRGQAkAm83djOqvoY8Bi4sZgOMpaAWRfXT
qp3ug365q9+yI3sMEsQ2dAt3o1ca4Iq0ATRHSgGjvNbpkBzCWIr4N/FIN1btu8AqttwJQhEJAE41zdirOcC
4NVwm7HOuriaLqHfaNcAX7Cy9bP7BVN+tqvuC6dqr04q0DupA2138LcUGCkiGbjEcBFuXut6IpIClHh3at
+O69Hkb7a3PKzUd3FNsS6uphOLiYOUUeHVUF1X5+IJ9s2NBgj8Tuq5uBLDflT1qqb2UdUar0vs27hurk+o
6ioRuRdYpqoLcVOVPiAiiqtiusHvPYfhSiD/CvTDdJSN9fdAWAnCdHJpmd78E2Fix0ao3O3u9DZBF2gV0z/
9nnfHNShva2knVV0ELGqw7C6/5/OB+U3sm4Nr6A47m4qti6vpItIyIXueG2wxHHpw5XtDfFsDdYcItIrpZf
/XIvI88HFQIooA1sXVdBn+c0OMmBnaWMA1UEfFuCk2TdC1dTCYkUCX7cKTU1xm1Uuma0jNdI8FK0M
bh0/eCtcNubPdmBimAm2DKGX/Noh83BwRXU5tnbK5uJzjrYur6Qp6JrvZDMOhoVrVVTGNDONJjDqZQ
KuYEoMdSKTI21VBVW0dQ60EYbqKcJkbojQfygqtgboDBVTFJCJni0hvv9d9ROSs4IUVvjYVuxE/rIur6TLS
Mt3wFtV7QxtHfQO1DfHdUQJtg7hbVXf5Xng3s90dnJDCm6+Lqw2zYbqMtPFuwqfCtaGNI28FIPsazk3Q
BZogGtuuSw5U4xvFNTXRuriaLsJXpRPqhuq8FW6MrDir8e4ogSaIZSLyOxEZ4f38DlgezMDCVU5xOcOSe1
oXV9N19M2Abj1D3w7hmwPCdJhAE8QPgCrgBWAesBe/u567kpziMobaLHKmK4mKcmMfhTJBVOyAnZ
ut/aGDBdqLqQy4LcixhD3r4mq6rLRM+Gq+62oailnc8rwGauvB1KEC7cX0roj08XvdV0QaHYK7M/N1cbV
pRk2Xk5YJlbvcVXwoWA+mkAi0iinFfxhuVd1BF7yTOqfIdXG1KibT5aR5d1SHqpopLxsSB4bHeFBdSKAJok
5E6qds80Za7bAJesJFTrF1cTVdVP+xIFGh68mUt8JKDyEQaFfVO4GPReRfgADH4k312ZXkFJXRvZt1cTVdU
Gw8JI0ITQmiqhyK18O4LnlvbkgFVIJQ1beALGAd8DzwE6AiiHGFpZziMoYmWRdX00WlZe5rC+hIBatA66
yBOgQCHazve8APcdOGfgkcCfyb/acg7fRyissZ0c+ql0wXlZYJq16Bip3Qo0/L27eX/BXu0aqYOlygbRA/BA4
HNqnqTGAysLP5XToXXxdXG+bbdFm+huqCVR37vnkroEdf6J3ese9rAk4Qe1V1L4CIxKnqWmBU8MIKP9
t2WhdX08WlhWhuiLxsV70UivsvurhAE0Sudx/EAuBdEXkN2BS8sMJP/SiuVoIwXVVCKvTs17HtELXVsH21
DbERIoHeSX229/QeEXkf6A28FbSowtBGr4urDfNtuiyRjp8bonAt1FbBgEkd956mXqunHFXVf6nqQlWtCk
ZA4WqTdXE1xlUzbV/rruw7gg2xEVJtnZO6y/HNQ21dXE2XlpYJtZVQtL5j3i8/G7rFQ/KIjnk/sx9LEAHaWG
SjuBrT4UNu5GW7aq2o6I55P7OfoCYIEZklIutEZIOIHDAarIgMFZElIpItIh+ISLrfuiEi8o6IrBGR1d7wHiFRW
6dsKamwHkzGJI+E6Dgo6IAEUVfnzQFh9z+EStAShIhEA48CpwBjgdkiMrbBZg8CT6vqBOBe4AG/dU8Dv1

HVMcARwPZgxdqS+i6u1oPJdHXRMdB/TMeUIHZshKo91oMphIJZgjgC2KCq33oN2vOAMxtsMxZ4z3v+v
m+9l0hiVPVdAFXdo6rlQYy1Wb5B+ixBGIM35MZKNzdEMOV5d1BbA3XIBDNBDAK2+L3O9Zb5WwGc4z
0/G0gUkWTgUGCniLwiIl+IyG+8EklI5Hj3QNgorsbgTtjlRVCaH9z3yc+GKK/EYkIi1I3UtwDTReQLYDqwFajF
3Z9xrLf+cGA4MKfhziJyjYgsE5FlhYWFQvSN4pr/8S4oL2HMREjbbx7DHY1U94Klxxi7P8uVIKZILYCg/1ep
3vL6qnqNlU9R1Un44YUx5uYKBf40queqsHdwX1YwzdQ1cdUNUtVs/r16xesz8Em6+JqzD6p49xjMBuqV
b0hNqyBOpSCmSCWAiNFJENEYoGLgIX+G4hIioj4YrgdeMJv3z4i4jvrHwesDmKszdpYVGbtD8b4dO8NfY
YGtwRRmueqsayBOqSCliC8K/8bgbeBNcCLqrpKRO4VkTO8zWYA60TkayAVuN/btxZXvbRERL7CTVL0t2
DF2hxfF9ehNsSGMfukZQY3QeTZEN/hINAZ5dpEVRcBixosu8vv+XxgfhP7/vguE/PLB18U1w0oQxuyTNgH
WvgFVZRAbhP+NvGzAG/vJhEyoG6nDnq+L61BLEMbskzYeUCgIUs1vfrYbXiMuITjHNwGxBNEC6+JqTCP
qh9wI0tDfvjkgTEhZgmiBr4trai/ramdMvd6DXWN1MCYPKi+BXZut/SEMWIJoQY7Xg0lsNitj9hGB1CA1V
PtKJdaDKeQsQbTAN8y3MaaBtEzXBlFX277HrZ8DwkoQoWYJohk2iqsxzUgbD9VlULKxfY+bnw29BkHP5
PY9rmk1SxDN2DeKq90DYcwBgtVQnbfC2h/ChCWIZtSP4molCGMO1G+0G0yvPRuqq8rcbHXWgyksWIJ
oRk6RDfNtTJNi4iBlVPs2VBesAtQaqMOEJYhm5BSX06NbtHVxNaYp7T3khs0BEVYsQTQjx5uH2rq4GtOE
tEw3sF5ZUfscL28F9EiC3uktb2uCzhJEMzZaF1djmtfec0PkZ7vqJbsoCwuWIJrguriWWwO1Mc1J9XoytUd
DdU0VbF9j1UthxBJEE7btrKC6VsmwYb6aNaVrPZEgc2D4liMK1UFtlXVzDiCWIJmwsslFcjQlIezVU1w+xYQ
kiXFiCaMIm7x4IG8XVmBakZULR11C99+COk5cN3XpC0oj2icscNEsQTdhY5Lq49k+0Lq7GNCttPNTVuCqi
g5Gf7Y4VZaelcGG/iSZsKrYursEExNeofDDVTHV1bn+rXgorliCasLG4zKqXjAlE3wxYKjJAlE3wxXNXQwPZlKvoWqPdaD
KcxYgmhETW0dW0rKrYHamEBERUHquIMrQeR7d1DbEBthxRJEI/J27bUursa0Rlom5K8E1bbtn7cCorpB
vzHtG5c5KJYgGrHRBukzpnXSxkPlLti5uW3752VD/zEQE9u+cZmDYgmiETbMtzGtdDAN1ar7htgwYcUSR
CNyrIurMa3TfyxIVNsaqndvg/Jim2I0DAU1QYjILBFZJyIbROS2RtYPFZElIpItIh+ISLrfuloR+dL7WRjMOBvKs
S6uxrRObLy7wa0tJYg8a6AOV0FLECISDTwKnAKMBWaLyNgGmz0IPK2qE4B7gQf81lWo6iTj54xgxdmY
HOviakzrpWW2bfrR/GxAIHV8u4dkDk4wSxBHABtU9k4wSxBHABtU9VtVrQLmAWc22GYs8J73/P1G1/P1G1nc4XxdXa38wppX
SxrtG6oqdrdsvLxuSD4G4hODEZdosmAliELDF73Wut8zfCuAc7/nZQKKIJHuvu4vIMhH5j4ic13dgbiMg13jb
LCgsL2yXobTtdF9dhydbF1ZhW8TVUF6xq3X7WQB22Qt1IfQswXUS+AKYDW4Fab91QVc0CLgb+ICIHjOC
lqo+papaqZvXr169dAqrvwWRdXI1pnTRvbojWtEOUl8CuLTbERpiKCeKxtwKD/V6ne8vqqeo2vBKEiCQA
56rqTm/dVu/xWxH5AJgMfBgpGrIF9CcLaIIxppYRUiE+BglYkCJuDOqwFswSxFBgpIhkiEgtcBOzXG0lEUkTE
F8PtwBPe8r4iEuZ2fpURHxtNP+viakzriLR+bgibAyKsBS1BqGoNcCPwNrAGeFFFV4nIvSLi6
5U0A1gnIl8DqcD93vIxwDIRWYFrvP6lqnZIgthU7MZgsi6uU7bQ6TLaOGbrGfRR1SQKfH8Ezwu11YFtn7cCeqVDfFJw4zJtEswqJl
R1EbCowbK7/J7PB+Y3st+nQGYwY2tKTlEZo9ISQ/HWxkS+tEyorYSi9ZDasFd7I/KyrfQQxoKaIEKturqa3Nx
c9u4NbKYrVeWOab1IiIthzZo1QY7OhFr37t1JT0+nW7duoQ6l8/BvqG4pQVTugeINkHle8OMybdKpE0Ru
bi6JiYkMGzYsoCqjyppavJLSe8bT1JPGzSsM1NVioLyc3NJSMjI9ThdB7JIyE6zmuovrD5bQtWAWoN1G
Es1N1cg2rv3r0kJycH3J5QVVMHQGxMp/5aDCAiJCcnB1y6NAGKjnGjsgbSUG1DbIS9Tn8mbE1jc6WXIOI
sQXQJ1hEhSHw9mVqaGyJ/BcQnQ6+G98+acGFnQj9VNXVEiRATZScOY9osLdONzlqa3/x2edmueskSddi
yBOGnsqaO2Jgou7I05mAEckd1TRVsX2PVS2HOEoSfqpq6dq1eKi4uZtKkkSUyaNIm0tDQGDRpU/7qq6
ZfZctW8ZNN93U4U4nscffTR7RXufs466yyOPPLIoBzbdHKp49a49xjMBuqV b0hNqyBOpSCmSCWAiNFJENEYoGLgIX+G4hIioj4YrgdeMJv3z4i4jvrHwesDmKszdpYVGbtD8b4dO8NfY

24O5Cjo+PZ/HixVx55ZXEx7v5M5KSWh5T58QTT6zfTlW54447mDBhAieccAJbt26loKCA9957j/PPP5+UlJ
T9jvu9732PuXPnAjB37lyuvPLKFt/PhEBaJqBQ0MgQanW1kL/ShtiIAJYgPB3ZxbVnz31XVP/3f//HzJkzWbl
yJa+//nqTN27Fxe0bXTY6Opqampo2bdOSF198kR07dpCRkcGwYcPIycnh+eefb/VxYmJiqKtzSbfhZ/L//M
899xyFhYUsX76cL7/8ktTU1GZvXps2bRo5OTl88MEH1NbWMn68TVMZlup7MjUyBWnJt1BdZj2YIoAlCE
+ourju2rWLQYPcjUJPPvvlkux9/1KhRfPvtt+Tk5ADwwgsvNLv9888/9888SePTksX76cefPmkZiYSH
p6OgsWLACgsrKS8vJyTjzxRObOnUt5x8wOQDz5x8wHmO9Xbt20b9b9/f7p168b777/Ppk
2bADjuuON46aWXKC4u3u+4AJdffjkXX3yxlR7CWe/BrrG6YOWB62wOiIhhCcLT3l1cA3Xrrbdy++23M3n
y5DZd8bekR48e/OlPf2LWrFmMmTKFxMREevfu2ei5Tk5bNq0ab/urRkZGfTu/u3ZvPPmmZZ555hocffpgE
yZw9NFHk5+fz6xZszjjDPIyspjupiqRJPPjggwDccsst/Pb9ZGzjjjDPIyspi0qRJPPjggwDccsst/PnPf2by5MkUFRU1Gd8ll1zCsmXLyMzM5O5Omnn2b06
NEAjBs3jjvvvJPp06czceJEfvzjH++3z9HJYEfvzjH++3z44dOwJqTzEhIgKpTTRU562AqG7Qb3THx2VaRbSl2+EjRFZWlvp6zfi
sWbOGMWPGtLivqrJy6276JcaS1rtHsEIMmT179pCYmEhsPNNzBy5EhuvvvnmUIfVZvPnz+e1117jmW
eeOehjBfo3Ytrgzdvg86fg9lyIit63/OkzoWH5rtaoOzokWIHXPTh6GIz7n22mtDPpgVLm
x1FNPUVVVxeTJk7n22mtDHVKb/eAHP+DNN99s0tZmAGPJFLW9sQittPFSXQ8lGSDnELVN190CMPjW0sZmA
WIJg3yiunXWQvptvvvmAEsPcuXN56KGH9ls2Hn300Y4MrdX++Mc/hjoEEyj/hmpfgtiVCxUl1oMpQ
liCoGO7uIaLK6+80hp5TXD1Gw1RMa6hevw5bpmvIVU
VPJuxZgsCVlOJsFFdj2l/DITfysyFlZON3V5uwYwkCV4LoStVLxnSYtPFQmgdlXlfnvBVWvRRBuvxZUVVDdg
+EMZ2e/9wQZcWwe6vdQR1BuvxZsaZOiY6SoHVxnTlzJm+//fZ+y/7whz9w/fXXN7q9/yio3/3ud9m5c+c
B29xzzz3/7nqyavrZ0i0O1nHVkcuD4/47Bcl2BwrN7RKcUWtbwU5ujVxUB2BaDnVwkhvc9jQ0Sq6fl0mmMNk
WWD9UOBJ4B+QAlwqarm+q3vBawGFqjgQcVzJu3NXpXZzdgLKAo0Mo2iLRMOOWXzW4ye/Zs5s2bx8
knn1y/bN68fAH09d/wYIFnHbaYwdOxaAe++9t83HaomZFVPkCFCo
JQkSigUeBU3Dn4NkiMrbBZG6+8DOuR2S2ltcgjQeeedxxvtvFE/g1xOTg33HMMPpLy/n/
ri3HjxnH33Xc3uu+wYcPqh6mm/+8C37a
pz9l0qRJfPPPNN8yZM6TKQlS5YwefJkMjMzMzueqqq6isrKx/v7a3bf4dVhgTAc9Ndd4KN0ZTf
Msj/powoapB+QGOAt72e307cHuDbVYBg73nAuz2WzcFmAfMAR5p6f2f2mTJmiDa1evfqFaZFw6qmn6oI
FC1RV9YEHHtCf/OQqKqqNTU1On36dxYxoWwqqk6fPl2XLl2qqqpDhw7VwoJCVsJCXbZsmY4fP17Lysp
0165dOmLECP3Nb36jqqpFRUX173/nn39vPUV4e3TfW6sCJHbhN
n+/ve/r38/3/6PPvqoXn311a1hn+rhNOOEE//PBDXbdunY4fP75
866igtKytTVa3//P6ft7CwUIcOHaqqnPztVBgwbVb1dd2du2q327EiBFaV1enK1eu1JEjR2phYeF+x5w5c6a+9u
0zZ46++uqrqr617/+VX/84x83+hnC5C5W+kU1v8M9V7+qr+brzq8xeHOhrTALBMmzivBrCwVCxz+R
fM3wrAu4OGs4FEEUkWgjt8AtdAK+aiZw1UDDmDmPy5MmsFrwVqv/7t
uPj6dWrF2eccceSmSvV9ZAoupUrVxccj7gfEedIk0dXBWV1x2EmPLI9lLC8GQL5rCoOdKPUprE9mhf9xsrFdIvT9suCrSgJfFiMkHP
CNsXkjTMDSMkFrYddmq16KMKFupL4FmC4iXwDTga1ALfA/wCL1a49ojhcIyLLRGRZYWEFh8KNtozpPPP
MlS5bw+eefU15eTlJ+SEg8++6CBliwOzububU6y8oivJdtdk5EJOozZZ84cHnnkEb2ESZSdi8ESZ
g/knBejBFlGAmiK3AYL/X6d6qeqyqPaqq6TVPUdaqaZvr1y9qIH+
PgJSQkMHPmTK666ipmz57N7N7N7t276dmzJ71796agoIBkkZg3WNBoIA333yz2f2/f2f2/853vsGDBAioqKigoKGDx4sWGhMIkhF8Mg
dx+4A+
AAQOorq7mueeeY6mWNpAccaNWoUOUOTk5nmMGNiNG6MBnNzmG6dOnWsRJSr2yvFPBf2fvHvlcwfgrtBBrZsBP0M0gFnXYbZsMbBF2
uDlzipgeVLbzQ+fsOsbzUv7tCbEGJ+fyCEjVPVxX2ClcPjOHCEzwJ4Hh+XGi9NPw0Uga7Pa4gyVqJ+LLSpbJNt2Dzoyi
D1iGuG0XDnLyKu9jSOB2nLvcyUO5bzKHxiiilHdJ9iOkc0Ckc1I8FKVqOS8kSLH76NqP9Dl8Myt2A4+
6E8Klc8SXt6SIE6dtBGQNRSRmsYAMEO8SESOw0eMXwQCX3cVchDtcwvoQt8E5TAHcC28SGFvMXUqSq44nEXqQgDU8K6bhBDvlSbvqnUVf
rNQoRMXwAQtSPqbFpzyPcJ4kbu1H6QaOa7lcfBXIXa1wMgDGp7I64DRGcPEWXPq9QD0w82D7GS7XTpTRRr9gVSwFumM14PWvOH5E7Wlc
n8aqBSt3aBJIPWMg4mnUVmRDp9wR6KMpE+SwcSK7ppsPNVKijRVEmmy1bKqDZWr8YFPb4IYq13qFAz5oDBgdfOMi3vPrjgRAHAbKvO8UP
rn7D8OL0pwVWvtO+DP8yMbChAT2O6j6yjJxaS9f0CrNaSxK7zwGVFEbAWIQwhqGDJi6NSIiWIhVMqn8nG6bSOLKYAbFD1MgZ+9l
tM9aDgXrDfGj9l1G0LK1LrqWJMjmbbMqbk36WMNPjClQ6F6WpZsAurmkxjqemAQ9MvzVPaoEGuuQoCzFRkfgyE9GYRGHgWyrKxbUT5E9LvX4sGdqrMtbRRoopJ8
gEWPyJ58fn6Vf6LQeo7VQB8mxWVrk8Qp7cHp+fkVqY3rNqdZk5UeeFyG2lQ5EiNqWSZb6J7OQbAbVyYcIgHbwi5Sfe8lUj8fpJrTk4
kpvOfzUAKqk+9GQmSdiK6xCuH9XIIK+Z+VtpvB4ynM9OzdqE4mznMZzAAxk3+Ndi2b9gfNZCDb9qKaLZL2GjpCuuiJKTHmJm4cm1VmC
eG2yG3yo8Fpqe5K8PjFtpD2nQIwHEdOG8XB6LBxSmkSyP+x+7xHPHu7Ahf1DqrWZWx5SLyqgCgiyPqqA+A09AsIUeMSD0R5Q0e3
ZTTB+A4yWU5qJw8ODH3zOfvEYWdhBHG4LMxvT8iLZnFZ3gVMK9LIFcXhvv3L+SuL11gCyJe1bHTpJQfD3Y2hGgaSE4O0HS4hGiuoJnS
SNz/2zcBxZjRR8GH56qNPuTWTBeNBeVY28vt/vM8LgHnz2+vLYy4a9fhwDxCwZDa7fKbGHZYHl9vjSE/5Blu5S3HgpvSgEtBmUE
Wg22ypZ/rKpP8fdYHQmPYCZrDqrvW/Gl3MxnMxCDsCRcNXrHyemr0Lq5FcvRH+iD8o6ZbP0DRFCSIgtNJtBrt9sY4ZL8vn/GjVyC9y+7HN
7L6t0FoTMMDzmzw4dDtYs5UgR3v6oZxBXgP/ndgt3mpcw3b5+T1Sc+HjHOv0o6JzC4ZaI0eXpmrZjtPqbdP1vpH+UqJGuQlVbhbeE4+0C
EP7AhI9gg8QxEq1iDMJt8CbAiYfdHdk6JR8wrkZHYmgy/m+hDq0DwJBDeAJPKqR4t1ydIT2C7IoBQzmJXq8wEVKEEumV5o4j0NZr7O
n4Aqd0Efr8x4+BUlPbiPDNPUvtPqrH8Gl/FLP3Fl86RnzmC3i4tVIt5YjCDZd8N9B9KZ31kmcr1o5zo0ZgVS0Dk4cB2Smpl/sRlYsSqXZZJ2Bb
PxHF8Lg2KBr+fa5s9J8fnRq4KIM7EHKUDdnGsYZ2ZbSsRZjGvy0VOxOw7NFPnPLqn6KYUdqMh4K/h1aqv2BmB6uB/Q2/xpiDVc
dYIeAP9OCuEnqYPIGXUxG9Cb3ue0bPQ1MVlHjIS/RSaZ7IqLk9m8SsNA/kx2rqgqOy5lSJKpPlpbeONOjlOyn+EDHmPsEBu0K
3Uc7N9zxczJPTV5hF3qVkE3oZWEnYtvZFT/KJkxUGhq9eAwEZJrIJNwzB6hTB5f7L8EjNkxBIoVYYoV3Sm7gAw0YwbSqL1Ov1YsNDI
6LzoImOqqu2n7V0WVNr5QP1Wv6Gg7rBz4XGBQwK3Q0yk2QOMa2P6o9ltFJ9bCl3njxPghOqmJK4+0X9tBdIjg0YM3wp3tpNfU/NQ
lgShq9q80hKWxp+VlW8yVnCgkDlthCyJUBI5rqzbULBaZzDDkdg6DqMNx/VX1+Np9gz6+rjL0SBgO0JGCZtXL9RaMGgsdkFOI1aR
0Dd4FLwYDXJ4K1iFwGVn4WvtQ1qo03Ln0NLX7tm0vKJ3Yk/HAMXcCCk+x0aq4gZfO2PSjmyIEuPvEBGQj3/q4D3oNB5G0Kw9Eg
DG2gk0aXYGpKv0Rfbh4QMGhEElq2uW7tfpBEvZOyNq3ZyJ6TBydLSK90gTqSQOZBvAH6/sbMpJsz1f+hPnh5S/CQE+5T7NcyZXhDzs
YEp6FUcWBTO6G1Tz+xFM/AvRSnNHrD1bzH5Mz/QF3cTlhAW

IrAl1EAF6CHhLVUcDEwnTuEVkEHATbire8bgpfS8KbVQHeBKY1WDZbcASVR0JLPFeh4MnOTDWd4Hxqj
oB+Bq4vaODasaTHBgvIjIYOAnY3F5v1KUTBHAEsEFVv1XVKmAecGaIY2qSquap6ufe81LcCWxQaKNqmo
ikA6cCj4c6lpaISG/gO8DfAVS1SlV3hjaqZsUAPUQkBogHtoU4nv2o6odASYPFZwJPec+fAs7q0KCa0Fisqv
qOqtZ4L/8DhM248E18twC/B24F2q3nUVdPEIOALX6vcwnjE64/ERkGTAY+C20kzfoD7g+2LtSBBCADKAT
melVij4tIz1AH1RhV3Qo8iLtSzAN2qeo7oY0qIKmqmuc9zwdSQxlMK1wFvBnqIJojImcCW1V1RXset6sniI
gkIgnAy8CPVHV3qONpjIicBmxX1eWhjiVAMcBhwJ9VdTJQRvhUgezHq7s/E5fUBgI9ReTS0EbVOur614d
9H3sRuRNXtftcqGNpiojEA3cAd7X3sbt6gtgKDPZ7ne4tC1si0g2XHJ5T1VdCHU8zpgFniEgOruruOBF5NrQ
hNSsXyFVVX4lsPi5hhKMTgI2qWqiq1cArwNEhjikQBSIyAMB73B7ieJolInOA04BLNLxvGBuBu1hY4f2/pQ
Ofi0jawR64qyeIpcBIEckQkVhcQ9/CEMfUJBERXB35GlX9XajjaY6q3q6q6ao6DPe9vqeqYXuVq6r5wBYRG
eUtOh5YHcKQmrMZOFJE4r2/ieMJ0wb1BhYCV3jPrwBeC2EszRKRWbjq0TNUtTzU8TRHVb9S1f6qOsz7f
8sFDvP+pg9Kl04QXiPUjcDbuH+wF1V1VWijatY04DLc1fiX3s93Qx1UJ/ID4DkRyQYmAb8IcTyN8ko584H
Pga9w/8dhNSyEiDwP/BsYJSK5InI18EvgRBFZjysF/TKUMfo0EesjQCLwrvd/9peQBumniXiD817hXXIyxhg
TKl26BGGMMaZpliCMMcY0yhKEMcaYRlmCMMYY0yhLEMYYYxplCcJEHG+0yt/6vb5FRO5pp2M/KSLnt
cexWnif870RY98P9ns1eN85IvJIR76niVyWIEwkqgTOaa8hjduLN3BeoK4Gvq+qM4MVjzEHyxKEiUQ1/7+
98wmxsgrj8PMM/aGFbgohB2rQanRlFNMqTFJsF21KrBCqRRoIBQUtIqhNmQTB0D8yM6lFVgulQIWwo
aZgrlQ2bWYii1q4SwIhJ2reFud89HW7zdgQ4x18H7jc757vfuc93wf3nPOew/39KH8Me6T7RHcGoJ6p7x
vUMfWgelJ9Vr1HnVAn1dWtajapx9XpqinV+FrsVjvVI+DBVr2fqIfo8c9rdWut/xt1Vy17ErgZeF3d3eOax1
pxnqplQ9Wf4O2aebxXNXhQN1aBwcnqFXBpLR9RP1NP1PtcVkOsVA9bfBmea93fvtrOSfUfzza58PgvM5
4k6SdeBL5uOrhzZB2wliKVfBLYExE3WYyXdgIP1+8NUaTgVwPH1GuAbRTV1JHaAY+rjYLqDRTvgO/bwdS
VwC7gRuA0cFS9IyKeVm8FHo2I413XbAaurfEFDqnrKfIaw8ADETGu7gUeqstF+4CNETGt7gd2qC8B7wBb
IqKjLgd+rWGupygBzwBT6iiwAhis/hLYRwY5yfkjM4hkSVJVbPdTjHPOlU711JgBvgOaDn6SMig0HIiI2Yj4lj
KQrKEYsWxTv6JIrF9O6cgBJroHh8oI8HEV1WsUQdfP08bN9fUlRUpjTSvOTxExXo/fomQhwxThvula/ma
NMQyciogOlOfV8jf4KCJ+iYizlKzn6nqfq9TRqkPUlyrByeKSGUSylHmB0om+0Sr7nTrxUQeAS1rnZlrHs63P
s/z9t9CtPxOU2fzOiDjSPqFuoEiD/18IPBMRr3bFGfqXdi2E9nP4A7goIk6r64DbgO3AXRQfhOQCJjOIZMkS
ET8DBygbvg0/UJZ0AG4HLl5A1XeqA3VfYhUwRRF03GGRW0e9zvkNhSaAW9QrLPa2W4Gxea45Atxv8fx
AHVRX1HNX+ZdP9t3Ap7VtQ3UZDIqY41gtv1IdqfUsm2sTvW74D0TE+8AT9K/UebKIZAaRLHWepyjyNrw
GHFRPAIdZ2Oz+R0rnvhzYHhFn1T2UZagvVCnuc3NaZkbEKfVx4BglM/gwIuaUuI6Io+pa4PMShjPAvZSZ/
hTFh3wvZWno5dq2+4B36wDQAV6JiN/ULcCoehll/2HTHKEHKW56zaSxnzyYk/NEqrkmyRKgLjF90GwiJ8
likEtMSZIkSU8yg0iSJEl6khlEkiRJ0pMcIJIkSZKe5ACRJEmS9CQhiCRJkqQnOUAkSZIgSVRORK5CYII=\n"

```
      },
      "metadata": {
        "needs_background": "light"
      }
     }
    ]
   },
   {
    "cell_type": "markdown",
    "source": [
     "###SAVE THE MODEL"
```

```
 ],
 "metadata": {
  "id": "Os2Z9O9lkK92"
 }
},
{
 "cell_type": "code",
 "source": [
  "model.save('Spam_sms_classifier.h5')"
 ],
 "metadata": {
  "id": "e9YPy-w-kMdN"
 },
 "execution_count": 15,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "###TEST THE MODEL"
 ],
 "metadata": {
  "id": "kkQtFl3KkSra"
 }
},
{
 "cell_type": "code",
 "source": [
  "test_sequences = tok.texts_to_sequences(X_test)\n",
  "test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)"
 ],
```

```
   "metadata": {
    "id": "1Z1VNZOvkVYx"
   },
   "execution_count": 16,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "accuracy1 = model.evaluate(test_sequences_matrix,Y_test)"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "LXeANR14kXlL",
    "outputId": "e2909723-5d54-4301-d210-c056b3f72b5e"
   },
   "execution_count": 17,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "44/44 [==============================] - 4s 87ms/step - loss: 0.1058 - accuracy: 0.9849\n"
     ]
    }
   ]
  },
  {
```

```
    "cell_type": "code",
    "source": [
     "print(' Accuracy : {:0.5f}'.format(accuracy1[0],accuracy1[1]))"
    ],
    "metadata": {
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "aJ5moLlCkZId",
     "outputId": "cc38b2fe-5e85-41ad-ae31-6171640bfa38"
    },
    "execution_count": 19,
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       " Accuracy : 0.10579\n"
      ]
     }
    ]
   }
  ]
}
```