```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": [],
      "gpuType": "T4"
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    },
    "accelerator": "GPU",

      "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": "markdown",
      "source": [
        "Unzip the Dataset"
      ],
      "metadata": {
        "id": "itwsJ09UMqDF"
      }
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "NV62eBgKzfZb"
      },
      "outputs": [],
      "source": [
        "import zipfile\n",
        "import os\n",
        "\n",
        "zip_path = \"/content/drive/MyDrive/LLM assignment 2/IMDB
Dataset.csv.zip\"\n",
        "extract_path = \"/content/drive/MyDrive/LLM assignment 2\"\n",
        "\n",
        "os.makedirs(extract_path, exist_ok=True)\n",
        "\n",
        "with zipfile.ZipFile(zip_path, 'r') as zip_ref:\n",
```

```
        "    zip_ref.extractall(extract_path)\n"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Install Required Libraries\n",
        "\n",
        "\n"
      ],
      "metadata": {
        "id": "SkaPy-7BNIFt"
      }
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ax-PPnvp1RD9",
        "outputId": "6c400917-7143-49a9-fe02-eb9eb0f5ddd3"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Requirement already satisfied: transformers in
/usr/local/lib/python3.11/dist-packages (4.53.1)\n",
            "Requirement already satisfied: datasets in
/usr/local/lib/python3.11/dist-packages (2.14.4)\n",
            "Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (1.6.1)\n",
            "Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.2)\n",
            "Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.10.0)\n",
            "Requirement already satisfied: tqdm in
/usr/local/lib/python3.11/dist-packages (4.67.1)\n",
            "Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)\n",
            "Requirement already satisfied: huggingface-hub<1.0,>=0.30.0
in /usr/local/lib/python3.11/dist-packages (from transformers)
(0.33.2)\n",
            "Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from transformers) (2.0.2)\n",
            "Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (24.2)\n",
            "Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)\n",
            "Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.11/dist-packages (from transformers)
(2024.11.6)\n",
            "Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)\n",
```

```
            "Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.21.2)\n",
            "Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)\n",
            "Requirement already satisfied: pyarrow>=8.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)\n",
            "Requirement already satisfied: dill<0.3.8,>=0.3.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.3.7)\n",
            "Requirement already satisfied: xxhash in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0)\n",
            "Requirement already satisfied: multiprocess in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.70.15)\n",
            "Requirement already satisfied: fsspec>=2021.11.1 in
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.11.1-
>datasets) (2025.3.2)\n",
            "Requirement already satisfied: aiohttp in
/usr/local/lib/python3.11/dist-packages (from datasets) (3.11.15)\n",
            "Requirement already satisfied: scipy>=1.6.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.15.3)\n",
            "Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.1)\n",
            "Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)\n",
            "Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)\n",
            "Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)\n",
            "Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)\n",
            "Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)\n",
            "Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)\n",
            "Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.5)\n",
            "Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)\n",
            "Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)\n",
            "Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)\n",
            "Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(2.6.1)\n",
            "Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.4.0)\n",
            "Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(25.3.0)\n",
            "Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.7.0)\n",
            "Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(6.6.3)\n",
```

```
            "Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(0.3.2)\n",
            "Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
(1.20.1)\n",
            "Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.30.0->transformers) (4.14.1)\n",
            "Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.30.0->transformers) (1.1.5)\n",
            "Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)\n",
            "Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(3.4.2)\n",
            "Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(3.10)\n",
            "Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2.4.0)\n",
            "Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2025.6.15)\n"
          ]
        }
      ],
      "source": [
        "!pip install transformers datasets scikit-learn pandas
matplotlib tqdm\n"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Import All Required Libraries"
      ],
      "metadata": {
        "id": "5eC6BqVkPGEI"
      }
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "yvXhJzQh1V8H"
      },
      "outputs": [],
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import torch\n",
        "from transformers import BertTokenizer,
BertForSequenceClassification, Trainer, TrainingArguments\n",
        "from sklearn.model_selection import train_test_split\n",
```

```
        "from sklearn.metrics import accuracy_score, f1_score\n",
        "from datasets import Dataset\n",
        "import matplotlib.pyplot as plt\n"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Load and Prepare the IMDb Dataset(test,train )"
      ],
      "metadata": {
        "id": "hKTxx5DYPOzX"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "from sklearn.model_selection import train_test_split\n",
        "\n",
        "df = pd.read_csv(\"/content/drive/MyDrive/LLM assignment 2/IMDB
Dataset.csv\")  # Load original data\n",
        "train_df, test_df = train_test_split(df, test_size=0.2,
stratify=df['sentiment'], random_state=42)  # Split data\n",
        "train_df.to_csv(\"/content/drive/MyDrive/LLM assignment
2/train_data.csv\", index=False)\n",
        "test_df.to_csv(\"/content/drive/MyDrive/LLM assignment
2/test_data.csv\", index=False)\n",
        "train_df, test_df = pd.read_csv(\"/content/drive/MyDrive/LLM
assignment 2/train_data.csv\"), pd.read_csv(\"/content/drive/MyDrive/LLM
assignment 2/test_data.csv\")\n",
        "print(f\"Train: {len(train_df)}, Test: {len(test_df)}\\n\\nTrain
dist:\\n{train_df['sentiment'].value_counts()}\\n\\nTest
dist:\\n{test_df['sentiment'].value_counts()}\\n\\nSamples:\\n{train_df[[
'review','sentiment']].head(3)}\")\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "WlxZcOInYB_r",
        "outputId": "9327a675-d65c-4079-8c14-f06efa483168"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Train: 40000, Test: 10000\n",
            "\n",
            "Train dist:\n",
            "sentiment\n",
            "positive    20000\n",
            "negative    20000\n",
            "Name: count, dtype: int64\n",
            "\n",
            "Test dist:\n",
```

```
            "sentiment\n",
            "negative    5000\n",
            "positive    5000\n",
            "Name: count, dtype: int64\n",
            "\n",
            "Samples:\n",
            "                                                review
sentiment\n",
            "0  I caught this little gem totally by accident b...
positive\n",
            "1  I can't believe that I let myself into this mo...
negative\n",
            "2  *spoiler alert!* it just gets to me the nerve ...
negative\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "class"
      ],
      "metadata": {
        "id": "jWbboPZxbLKl"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "# Print unique class labels\n",
        "print(\"Class labels:\", train_df['sentiment'].unique())\n",
        "\n",
        "# Print label distribution in train and test sets\n",
        "print(\"\\nTraining set label counts:\\n\",
train_df['sentiment'].value_counts())\n",
        "print(\"\\nTest set label counts:\\n\",
test_df['sentiment'].value_counts())\n",
        "class_labels = sorted(train_df['sentiment'].unique())\n",
        "print(\"Class label list:\", class_labels)\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "HQKyvmOtZsz5",
        "outputId": "41ef2987-e201-4f2b-ccfb-fe0c6cdf0113"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Class labels: ['positive' 'negative']\n",
            "\n",
            "Training set label counts:\n",
            " sentiment\n",
```

```
            "positive    20000\n",
            "negative    20000\n",
            "Name: count, dtype: int64\n",
            "\n",
            "Test set label counts:\n",
            " sentiment\n",
            "negative    5000\n",
            "positive    5000\n",
            "Name: count, dtype: int64\n",
            "Class label list: ['negative', 'positive']\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "EDA\n"
      ],
      "metadata": {
        "id": "-QfIHKptawFH"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd, seaborn as sns, matplotlib.pyplot as
plt\n",
        "\n",
        "df['length'] = df['review'].str.len()\n",
        "\n",
        "# Bar plot with counts\n",
        "plt.figure(figsize=(6,4))\n",
        "ax = sns.countplot(x='sentiment', data=df, palette='Set2')\n",
        "plt.title(\"Class Distribution (Full Dataset)\")\n",
        "for p in ax.patches:\n",
        "    ax.annotate(f'{p.get_height()}', (p.get_x()+0.3,
p.get_height()+500))\n",
        "plt.legend(title=\"Sentiment\",
labels=df['sentiment'].unique())\n",
        "plt.show()\n",
        "\n",
        "# Review length histogram\n",
        "plt.figure(figsize=(6,4))\n",
        "sns.histplot(df['length'], bins=30, kde=False,
color='skyblue')\n",
        "plt.title(\"Review Lengths (Full Dataset)\")\n",
        "plt.xlabel(\"Number of Characters\")\n",
        "plt.ylabel(\"Frequency\")\n",
        "plt.show()\n",
        "\n",
        "# Print one example per class\n",
        "for label in df['sentiment'].unique():\n",
        "    print(f\"\\n{label} sample:\\n\",
df[df['sentiment']==label]['review'].iloc[0])\n"
      ],
      "metadata": {
        "colab": {
```

"base_uri": "https://localhost:8080/",
"height": 1000
},
"id": "w4TGdLIHbleX",
"outputId": "d702a573-bdfe-423b-add4-1f38a933debc"
},
"execution_count": null,
"outputs": [
{
"output_type": "stream",
"name": "stderr",
"text": [
"/tmp/ipython-input-7-2603806220.py:7: FutureWarning: \n",
"\n",
"Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.\n",
"\n",
" ax = sns.countplot(x='sentiment', data=df, palette='Set2')\n"
]
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 600x400 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAi4AAAGJCAYAAACtu7gUAAAAOnRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjEwLjAsIGh0dHBzOi8vbWF0cGxvdGxpYi5vcmcvlHJYcgAAAAlwS
FlzAAAPYQAAD2EBqD+naQAAVRlJREFUeJzt3XdUFFf/BvBnBXepu1ioioBoFBQxWBB7FMWWaG
KPscUSDaiIsZBY0Gj4hcQWa4oRk9g1ahQLiAULNuyNKEHHRKGIDBKTf3x++zOsKKCC4jO/zOWf
PcWbu3PnOLCsPM3dmFUIIASIiIiIZqKDrAoiIiIiKisGFiIiIZIPhYiIiGSDwYWIWIihkg8GF
iIiIZIPBhYiIiGSDwYWIiIhkg8GFiIiIZIPBhYiIiGSDwYXoBfb29hgyZIiuy3htAQEBUCgUb
2Rbbdu2Rdu2baXpAwcOQKFQYNOmTW9k+0OGDIG9vf0b2VZBTpw4s3b5ZJ/wXtn0KhQE
BAQJls723y8OFDGBsbY+fOnbouhUoJgwv9z4iJicFnn32GmjVrwsDAGq1Gi1atMDChQvx9Ol
TXZf3UsHBwAoFNLLwMAANjY28PLywg8//DgIAAnD17tlT6K03lubavvvoK/y9n7s3
LmzwCBXpUoVDB8+HNOmTXvzRVGZ0Nd1AURvQkhICHr37g2VSoVBgwahfv36ysy4k
RcunQJP/30k67LfKVZs2bBwcEEBWVlZiI+Px4EDB+Dr64/QoMGDaS2U6dOpU4r
V/507dzBz5kzY29ujYcO9ujYcOGRV4vNDS0WNspiZfV9vPPPPyM3N7fMayjI2Nfsu
qyMwMDDffBsmzdRmqR///7o0qULcnNz8fjxY5w8e8eRILFizAwoULsWLFCvTr16/YfZ0Z6Ws7
Ny5E0uWLCkwIwaFPo//PAD9u3bh3bt2r354qhMNmmZKAzgI9Oz//GA3+rP1XhpdLeZqPYV8vsvwthgddPZYSFvwjgvYv9t6EVvbMydmgdcS7tVDfhuVVNlZOmJwRKPJ4KhmMIOFaFYb/Afbu8FTgHgRxxUMy/LFMq4gUVBIxgPAM+v1FVx8zK4ZA+a8sbN3TqBZT3gA+2vuaEQZgNjmhPSZHo8dlsV4E/8Cp04Vm9g5Ywpy6Dq5JA4AAMKEwDLg/DAWGgmyAxmHL1VVh0h7PkqN7H3iZZ7rV0JFGw+o8bhJjGZzO8bG/9tq1jZsgI1j1zhy3yKp2Q4tjKU2tsX27g0QA4NrtNXiufy2RaZB3nVWHswW0nAjPg71FC0RDA0tBbk0lHfrafn8DbkkJb7t6aa4Z6gY8p1Wo7gnQLeSqpaQZy7bJN49CVI12OfVmwYRaoTIYWAY0pp8/ERMY6NNdc1ny6yl0V8tB2ILGsyzN4y4QDSq5AMnJoeMlUfxQUlW8bIKfSqVIvF2oJMCO/vzPy/Dz5Tt7yK/G/JLXKwdCHEIU55iG9UQLKaX9ppKuDyKBVE0w1ZcPHaUFjU62P19/TCovyXfESMDZkGw6eFn5m5wd2NydH+hO0hWIwlvF7U5ilSqh2ph5Y1u0MzuNO4afPp3t1Ka91JLQpq/LbL0i/pPd0d/1ot7E20xCj/Utr2qBbiFSLi0NLMjc9iiwtMXM/LSW+CqGEXIA/KPVJYPYP8bJv/R2+g07fuphCZg9oG/jQXmZ1CQUW6E4x2SBdqR0B89kG0wnoBhJoZyEKb4h9a2qcSl5/y/i3LXwVGJp5Ds9nc3R2gxHt0cn+TwffFaR6XxYpfEFtpBuh2zHaqcwvswehrCdQ5ruD7TtIcNfcVzvE3wEO3cvXjWsXH7/7DVVJuaRlnnrz0Xy9E2tLv1xqAPBS1gtCV+Jt6GTE4pPL2SPSj+NYUL2uAp6w0p1gOlS6f8PO7cAUWSJ2FoUGDnUKEJ6qHZ6IuIJBUY6dE0y6+8smJz2pkv1QbUjfBTRxBpbLaaBo4TiggOoezaXKfBpN2Cuz8KSC9vm+T/GTb4VM1DnoV5ZScQtNlMqaugw6Ls1kyUrAqdKD86mGD/KeKwFYpXRoAqMG4vegVkeEk8SC9AkCXHdQJwXJV6fYfb9XqMeEECkRGWmIhV3PnmXHvyCHt67BMZJoJLEcfdOlFlSF7fX0HxJzLKtc/UiSeh5oKgOApT82QdyHzq6EaphXEHG6hD8+tugAbcM48NTSTz3rhr4H/pXA9pUb3s85qxpAbkzP0Er1F9p0YN05P/mi6unYYCbBSRNTeBT8qzdLEr8fC4+60MgwpW8UsfNftIIwvVsAeRwnoMlfAmapeuM0Yfog7rbBDoDaPG3Lme2sIH1tQ8JsYBsCEJPewNoiCKyNoeiDKDpEF4Bs5kbHE3KqSfG51Ka+UyWwbdyv9QJ52tfeBAHYqBDvHS2wsnVc+K7g3SKNxL+8rMdIBlsoiXEhgjgjOakwMatX5C
KPMiO/CAq7xlhNMMBRrNWFD
-->

Z84cKBQKTJ48GQkJCViwYAE8PT1x9uxZ6cxQURSltucJIfDBBx9g//79GDZsGBo2bIg9e/Zg4
sSJ+PfffzF//nyt9ocPH8aff/6Jzz//HKampvjhhx/Qs2dPxMXFoUqVKoXW9e+//yIuLq7Q9y
4nJwcPHjzQmmdgYAATE5Mi73tZ8vDwgKOjI8LCwqR5Rfl5f9X7sXHjRqSlpWH06NGoUqUKTpw
4gUWLFuH27dvYuHGjtK2ePXvi0qVLGDNmDOzt7ZGQkICwsDDExcVJQfT333/H4MGD4eXlhW+/
/RZpaWlYtmwZWrZsiTNnzsDe3h6fffYZ7ty5g7CwMPz++F7mujRo0wf/58XLp0CfXr1y+jI
0pvhCB6iyUlJQkAonv37kVex87OTgwePFiaTk9PFzk5OVptYmNjhUqlErNmzZLmde/eXdSrV+
+lfWHs0GuHt7V3kWvKsXLlSABAnT558ad/vvvvuuND1jxgzx/Ed8/vz5AoC4f/9+oX2cPHlSABA
rV67Mt6xNmzYCgFi+fHmBy9q0aSNN79+/XwAQ1apVE8nJydL8DRs2CABi4cKF0rwXj3dfb6s
tsGDBws7OztpeuvWrQKAmD17tla7Xr16CYVCIa5fvy7NAyCUSqXWvHPnzgkAYtGiRfm29by9e
/cKAGL79u0F1g8g3ytvX/Pe09jYWK318o7d/v37C92/vLpnzJjx0vpiY2MFAPHdd98V2qZ79+
4CgEhKShJCFP3n/WXvR1a4Wr55gYGBQqFQiJs3bwohhHj8+PEra3vy5IkwMzMTI0aM0JofHx8
vNBqN1nxvb2/xsl9pR48D8sBLRfRWS05OBgCYmpqWuA+VSoUKFZ59VHJycpKYmqXWuA+VSoUKFZ59VHJycvDw
4UPpMsvzl3jMzMxw+/ZtzMcPz4cdy5c6fE9RTGxMTkpXcXmZmZAQC2bdtW4oGsK
pUKQ4cOLXL7QYMGsQpeuvxYrfkTJkyAEAK7du3Smu/p6QlHR0
dpukGDBlCr1fjnn39eup2HDx8CACpVqlTgcnt7e4SFhWm9Jk2aVJJdKjN5Z3/yfnaK+vP+Ms+
fTUtNTcWDBw/QvHlzCCFw5swZqY1SqcBAwfw+PHjAvsJCwtDYmIi+vfvjwcPHkgvPT09uLu7
Y//+/UXez7z6MUzYCQ/DC70VlOr1QDwWrcL5+bmYv78+ahduzZUKhWqVq0Kc3NznD9/HklJS
VK7yZMnw8TEBE2bNkXt2rXh7e0tXYbJExQUhHIsXL8LW1hZNmzZFQEDAK385FlVKSspLA1rfvn
3RokULDB8+HJaWlujLujXrx82bNhQrBYFArVq18o1hKG03b96EmSrVuzERV27tta0QqFArVq18o1hKG03b96EmSrVuzERV27tta
h5OQkLX9ejRo18vVRqVKn+hvkgUMm7C2NgYnp6eWi9nZ+ci9fmmpKSkAPhvuCn6z/vL/p5i4CF3urr+qtumbuXJs4b
YciQIahcuTJMTExgbm6ONm3aAHDUh0qlwrfffotdu3bB0tISrVu3xtatW/ONdwOA1q1b51lWrk5Nz48T5C/qFr3WcmRRUr3MUtR7r
C5zc3OtV2hoKBISEoq8nvHHVHQYXequp1WrY2Njg4sWLJe7
gDe/bsQVhYGOrVq6f1S9/JyQRd0dFYt24dWrZsiTNnzsDe3h6fffYZ7ty5g7CwMPz5559/sGjRIt0wFn
Y2OC7775DvXr18p0BKK7bt28jKSkJtWrVKDzjnf7WzthjPPZ1TbnNnc9H37590aH
5BRpO8UZl1JUhf0iKWpNpUFPT6/A+YUFkjjxQX9ee9MDk5O
ejQoQNCQkIwefJkbN26FWFhYdLA3ef78PX1xd9//43AwEAYGBhg2rRpcHJyks7K6LX9/fff85
25CgsLw7Zt24q8n3nvUdWqVYu8DpVPHJxLb71u3brhp59+eIDkfKJpb3bvd0/qZm/Dee+9hxxsY
VWvMTExPz/SdobGyMvn37om/fvsjMzMs2NnwN/fHwYGBgAAa2trfP755/j888fP755/j888+RkJAA
Nzc3zJkzB507dy7xPuNSPTy8npPuwoVKqB9+/Zo37495s2bJ2+++QZTp07F/v374enpqfBa+g/8XO35B8
W9jrp16kvJ4dst9ceVdunhx38vq6bsFiYyMRExMjNat0kX9eS/bhVCoMGDZLXCa5KnnYlnUMK9MffbeOBE/fuxZMnT7TOuQB/O35B8
W9jrp16wJ4dst9ceVdunhx38vq6bsFiYyMRExMjNat0kX9eS/s/bhw4QL+/vtvrFq1CoMGDZL
mPz8A+HmOjo6YMGCu/vjjD+nynYWFBTw9Pf6L6/6+ch7j/LOupF8
8YwLvfUmTZoEY2NjTB9NjDB8+HPfu3cu3PCYmBgsXLix0/T09Oz+qxLPHuCDmKYDsQvVcSrfVoh1UJqGxf/Wcvzv4u0nXz8zlkOZdJR86Q
J2dIYRAVlYWcnJy8p1qt7CwgI2NDTU1WTTIyMoq7SK6Yp6bs/eVHHu+WUi9YXHhAWZbZ/eFOCWt5I6T+hLL4SQf1C1fnS6J5N0F8Si
ztGxsbA8j/y7SkfvvtN63LdJs2bcLdu3e1QpqjoyOHTuGzMxMad6O7RGhRcvNtb+UmVxw29eUFQ6C6dN3pg5utbJ3rS+G0mR5/y3NQ
cLF68WGv+/PnzoVAoXiskPq9atWxMfDp5Nr1VWCE+u3i/UxQmjGXmMpE5pF/BYxWN3JXBmdHdFWqP33k8d6Dhj+6MGhx9xn+VyFuRPkx6u8YBHVfo6eA
SkycOFGaX9Sf98Lej7yzSzQ1LS0tDWd+KMjyh2QNLGsjJ7M8VHbwIXXKqJn+SFx0b5q9j6QjBHqM1bREAfHrTxLqGA5
W8f7t+//8p68kRFRUGj0aBevXoFLif54BkXeus5Ojp1jBuu0kmSzMuT8ccurd9OeZXHmdWMszEqOuiaRRPQrzI
9N1K1bN8yaNQtDhw5F8+bNceHCBaxZs6ZIY38OYwu5eebqVgRQgu6v3h2BtwaiaXUhc3pqSEY18y8p5vezntsTPmeRbzlmbNxOnY8YYfvk7E7e03w
pqSkSExNRvXp19OrVC66urvfHAaEJAbLxhaYWM1QCjjdnrrsrJxEmvjD9mqcaQ59Qzh8qX98oXgfCm4oxqFMWXZt8vVYcLxUxzV0Gg1eCnJW
MNjZ2eGvv/6SzuoUZNasWYiIIEDXrl1hZ2eHhIQELF26FNWrV0fLli2lY2VmZ0bly5fD1NQUx
sbGcHd3h4ODQ5Hqe1HlypXRsmVLDB06FPfu3cPHPfu3cOCBQtQq1YtrVu2hw8fjk2bNqFTp07o06cPYm
JitP7azlOc2t5//3289957/Oqrr3DjxjxuroiNDQU27Ztg6+vvb76+X0f37t2xZcsW8B1WCCGHkdVa
oXr16tPn/fHo0SNUrlwZ69atq4JuXjy+GnHtaT6J4/kDRq4dXhN5C3pQ0D3nE+N0QXnAxOFR8YkDkSAvPX6INeX
77//rnUGrKg/74W9H1oXjoyO++OIL/Pvvr1di8eXO+y2l///032rdvvjz59+sDZ2RaL/afCC5Km7cLE/A79v3Rn6+
vrYsmmaltJ3Wz3JV61WY9ohmyzMccHS7dZ4ND3NczcN04mRrNlJGHOdbtTR1i/DmDJq8eaIYvLm4uDDXmlChAIdzd9P
y8vKCnp6gf1ROCwsDC8//77HOPyNtDNUxEbffG/8tRowYIaFt7YVSqRImcqGqcQSUmfGaiBSYWVtGiRIS
9PV1qV9Dt0BMmTBDW1tbC0NBQtGjRQkRoGkZRkGGra7XffHH38UrVu3FlWqUo4OsjiqKrvMzMDTd2ZAS
ZmRkiIkTJwpXV1dhamoqoqOakjI2Nhaurq6dOkra8+7dTbvpVQhZWvlEjQoYNYuHCh1i3HeV6H8H
To8PFx0795d2NjYCKVSSWxbET//v3Fe3Xr1X9Xe63bVNmzaF3u5d2O3Qa9
euFf7+/sLCwkIYGhqKrl27SrfDPm/u3LiWrVqVqSiRYtWohTp07l6/NltRV0u/CTJ0/E+PH
jhY2NjahYsaKoXbu2+O6777RuHRbi2S3C3t7eBjc2w1T3z0rMzhzN5eHasqsbwe3f2e
VCqVsLS0FF9++aUICwsrljZj+g7ud8h178o+vqicuXKwt3dXfj7+xf4Xo9PT2FiY
iKqVq0qRowYIaFt7YVSqRImcqGqcQSUmfGaiBSYWVtGiRIS
AQjo6OYsiQIeLUqVNSm+zsbDFmzBhhbm4uFAqF1q3jxfnaVprzD+/r4OcbrSPmfQQzd0UDb
fvj1sbGwKffgZ6Zavry8iIiIQFRXFMy5vAQYXIqLXdPz4cbRq1QrXbVXKGFyIiHNnhXERERERck
GgwsRERHJBoMLERERyQaDCxEREckGwwsRERHJBhMLERERyQaDCxEREckGH0BXSnJIYIF0v
3Hnzh2Ymprydrs3bO7cudi+fTu+ivVffh8Ne1a1ccPnxYa72hQ4diwYw7xawhqF0v

StW7fg5+eHQ4cOwdjYGP3790dAQAD09f/7MTl06BC+/PJLXL16FdWqVcPEiRPzPa32559/xg8
//IB79+6hfv36+O6776SHYxVmy5YtmDNnDuLi4uDo6IiZM2eiY8eOr3FUiMoPfkbpVYQQePLk
CWxsbKRvJy8M7yoqJbdv34atra2uyyAiIpKtW7duoXr16i9twzMupSTvi9xu3bolfcsq6caDB
w/g6OiInTt3okWLFgCe/TXn4uKC//u//ytwnbCwMPTp0wfR0dGwsLAAAKxYsQIBAQGIiYmBUq
nE9OnTERoaimPHjknrDR06FElJSfjzzz8BAO3atYObmxu+//57AM/OxDk7O2PkyJHw8/MrcNt
DhgxBWloaNmzYIM1r3749XXFxctP7aJHpb8DNKL0pOToatra3Wl6IWSmfP7H3LJCUlCQDS491J
d65duyAiAsXLkjz2rRpI6pWrSqqVKki6tWrJ6ZMmSJSU1Ol5dOmTRourq5a/fzzzzz8CgDh9+
rQQQohWrVqJcePGabX59ddfhVqtFkI8e6S/np6e2LJli1abQYMGiQ8++KDQem1tbcX8+f015k
2fPl00aNCgiHtMJC/8jNKLivM7lGdc6K2Sm5sLX19ftGj5fmf/zxx7Czs4ONjQ3Onz+
PyZMnIzo6WvorLD4+HpaWllp95U3Hx8e/tE1cjKePn2Kx48fIycnp8r8A2V69eLbTmwvrN2y7R
24SfUXpdDC70VvH29sbFixfzDfIbOXKk9G8XFxfdYW1ujffv2iImJgJKdVvCiail+NnlF4Xb4emt
4aPjw927NiB/fv3v3Jwl7u7OwDg+vXrAArKyvcu3dPq03dq03etJWV1UvbqNVGBoaomrVqtDT0y
uwTV4fBSms35etQyRH/IxSaWBwIdkTQsDHxwdbtmzBvn374ODg8Mp1zp0DCwCwtrYGAHh4eOD
ChQtISEiQ2oSFhUGtVsPZ2VlqEx4vertVPWFgYPDw8AABKpRKNGjXSapObm4vw8HCpTUFe1S+R
3PEzSqWqzEfc/I/g4FzdGT16tNBoNOLAgQPi7t270istU0tIIcT169lfFrFmzxKlTp0RsbKzYt
m2bqFmzpmjdurXUR+ZR3Z2tqhfv77o2LGjOHv2Rv2rNi9e7cwNzcX/v7+Upt//vlHGBkZiYkTJ4orV6
6IJUuWCD09PbF7926pzbp164BwcHi8uXYuTIkcLMzEzEx8dLbQYOHCimTJkiTR85OcrFsdE85ckT
o6+uL77//Xly5ckXMmDFDVKxYUWvgIpGvvgIpGc8TNKr1Kc36EMLqWEwUV3ABT4WrlypRBCiLi4ONG6
dWtRuXJloVKpRK1atcTEiRPzvVc3btwQnQTt3FoaGhqJq1apiwQJIsrS6vN/v37RcOGDYVSq
RQ1a9aUtvG8Rz8RYsWiRo1agilUimaNm0qjh07prW8r8W8TZs2YvDgwVrzPFvXr1RE
hIyGsfF6Lygp9RepXi/A7lA+hKSXJyMjQaDZKSkvgcFyIiomIozu9QXpNjdPIO5J0pcJtq5daaD4cO3YMYWFhyMrKQ
g0qlQq1atRAcHJyvniVLlsDe3l56JPWJEydKf2Z+HQaXA4ePAhvb28cO3YMYWFhyMrKQ
seOHZGamqrVbssSIEbh79670Cgo0Kkpbl5OSga9euyMzMxNGjR7Fq1SqEhIRg+vTpUpvY2Fh07d
oV7733Hs6ePQtfX18MHz4ce/bskdqsX78eEyZMQIBAQ2Ey8NkFvfdXMaaDnB4ki3SpkBhEQXY3k
u378PCwsLHDx4EK1btwdwtWzfcuXNHekDQ8uXLMXnxyNy/fx9Kp0RKKRKeThiSpPPKBXl4Ij
J09GSEgILl68KK3Xr18/JCYmYvZ8GMCzkea2trYYM2paykea2trYYM2YMpkyZ8sraseamIi
IiozGRzqehFSUlJAIDKlStrzV+9ejWqVq2K+vXrw9/fH2lpadKKyyMhIuLi4aD3V0MvLC8nJb
h06ZLUxtPTU6tPLyy8vREZGAgAyAyMzMRFRWl1aZChQrw9PSU2rwoIyIyMDycnJWi8iIiqW+Xmybl
yewx0YGAgZs6c+Xo7XQwTdv32xrZFpCtzOw/SdQkllrBskq5LICpFqZ4LICpzFqODXt2ojJWb4CK3x0D7
+/trfZNo3jdbEhERUdkpF8El7zQHQMtKOjI6ysrPLd/VPcx0Dr6ekV+zHQKpUKKpWqWq6
DtJRERERr02nY1yEzB8DTURERG+WTs+4eHt7ozaNdi2bRtMTU2lMkQaGhoaIiYnBmjVr0K
VLF1SpUgUgXnz5/H+PHj0bp0bp1azRo0AAA0LFjRzg7O2PgwIGIEcIGpCfHw8pk6dCm9vb6lh
48WJMmjJQn376Kfbt24eNGzdi586dUi0TJkzA4MGD0bhx09bRpUyrACpqakYOnTONomz8wRERE
VCCdBpdly5y5YBeHbL8/NWrlyIBQdWlyJIUOGQYu/evVKIsLW1Rlprz6Hnbs2IHRo0fDw8MDJiYmGGGG
4eDRs2x07du/MN2CUiIiLdKVfcPcZZGzsn6OC+8qov8FvKuIqHwrq7uKZPscFyIiIqKXAhIiI
i2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEi
IiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwI
SIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYH
AhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItl
gcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi
2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItlgcCEiIiLZYHAhIiIi2WBwISIiItnQaXAJD
AxEkyZNYGpqCgsLC/To0QPR0dFabdLT0+Ht7Y0qVaqgUqVK6N27N+7du6fVJi4uDl27doWxsTE
sLCwwceJEZGdna7U5cOAA3NzcoFKpULt2bYSEhOSrZ8mSJahRowYMDQ3h7u6OEydOlHifiYiIi
KFhQUmTpyI7OxsrTYHDhyAm5ubVCoVateuXWDaJ079xsrTYHDhyAm5sbVCoVatWqheDg4HDyNwPJ
EJafT4HLw4EF4e3vj2LFjCAsLQ1ZWFjp27KhQ1ZWFjp27IjU1FSpfjx47F9+3Zs3LgRBw8exJ07d/
y3NyctC1a1dkZmbi6NGjWLVqFVqCDgzF9+nSpTWxsLLp27Yr33nsPZ8+eha+vL4YYPH449e/ZIb
davXw8/Pz/MmDEDp0+fhqurK7y8vJCQkKBQ8vJCQkPBmDgYRERG9kkIIIXRdJ779+/DDwsICBg4c
GUlARzc3OsWbMGvXr1AgBcvXoV9evVk5MzNzaWMWfsmdf/+4069YNd+7cgaWlJQBg2bJlmDx5Mu
fvw+lUonJkycjNDQUFy9elLbVt29fJCYmYvfu3QCeMVvz81A8bqWZ/PAcgpsJwi+wZhu6ua9AJW6bnIWl5zYy1zmUpxjAD4
+lUonJkycjNDQUFy9elLbVt29fJCYmYvfu3QCeMVvz81A8bqWZ/PAcgpsJwi+wZhu6ua9AJW6bnIWl5zYy1zmUpxjAD4
KVNeWXtycjI0Gg2SkpKgVqtL+9Bgwq7fSr1PovJmbudBui6hxBKWTdJ1CURlzmJ0UJn0W5zfo
eVqjEtSUhIAoHLlygCAqKgoZGVlwdPTU2pTt25d1KhRA5GRkUDSkKQCAyMhIuLi4SKEFALy8vJCcnI
xLly5JbZ7vI69OrorSalOhQgV4enpKbV6UkZGB5OTkrV5ERFQs+qQkbV6UkZGB5ORkrRcRERGVrXITXHJzc
WLVqgfv36AID4+HgoFApUqlRJq521tTXi4+OnNk+vVPFHh6g+aMrwXoVmT48Pc5OhQKzaqvrjJsbbWJyceJN7zDSz5ex9d
18eLAgMDodFopJetrW3dpyIiIKrNwEF29vb1y8eBHr1q3TdSnFNXW/bw9+3Mw5fd0rHv5HUq
ET01tPXdQEA4OPjgx07diD6djT5cw0j39+6/jqOP+SxbXW1hrn57+2Ke1Yt7r8DdJSbdaeXaNk6v0NZk9+0nP/PWIqWNEqd2KH3QH/8rFn//VHUKdXgaqYxfJjOkKs6Eq2ZjV8WxuRkUNjuFXn3zXUfSmj17LDk
sX70S6d+8e1Go1DA0NoaenBz0zPoOaQNVIhoXv4bsQQ01vQLb5PXxIpVKBZVKVbIdJiIiohLR6RkXhJ+4hbdI8HGFyoVhSWZ7nz
3Dw4ODlrLGzVqhIoVKyI8PPFyaFx0djbi4OHDx4EAA8PD1y4cAEJCQlSm7CwMKjVajg7O0tt

vDZ5fSiVSjRq1EirTW5uLsLDw6U2REREpHs6PePi7e2NNWvWYNu2bTA1NZXGk2g0GhgaGkKj0
WDYsGHw8/ND5cqVoVarMWbMGHh4eKBZs2YAgI4dO8LZ2RkDBw5EUFAQ4uPjMXXqVHh7e0tnRE
aNGoXFixdj0qRJ+PTTT7Fv3z5s2LABISEhUi1+fn4YPHgwGjdujKZNm2LBggVITU3F0KFD3/y
BISIiogLpNLgsW7YMANC2bVut+StXrsSQIUMAAPPnz0eFChXQs2dPZGRkwMvLC0uXLpXa6unp
YceOHRg9ejQ8PDxgbGyMwYMHY9asWVIbBwcHhISEYPz48Vi4cCGqV6+OX375BV5eXlKbvn374
v79+5g+fTri4+PRsGFD7N69O9+AXSIiItKdcvUcFznjc1yIXh+f40JUvvE5LkREETFwOBCRE
REssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJ
ERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHg
QkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyw
eBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERL
LB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCRER
EssHgQkRERLLB4EJERESyweBCREREssHgQkRERLLB4EJERESyweBCREREssHgQkRERLKh0+AS
ERGB999/HzY2NlAoFNi6davW8iFDhkChUGi9OnXqpNXm0aNHGDBgANRqNczMzDBs2DCkpKRot
Tl//jxatWoFAwMD2NraIigoKF8tGzduRN26dWFgYAAXFxfs3Lmz1PeXiIiIXo9Og0tqaipcXV
2xZMmSQtt06tQJd+/elV5r167VWj5gwABcunLJYWFh2LFjByIiIjBy5EhpeXJyMjp27Ag7Ozt
ERUXhu+++Q0BAAH766SepzdGjR9G/f38MGzYMZ86cQY8ePdCjRw9cvHix9HeaiIiISkxflxvv
3LkzOnfu/NI2KpUKlVlZWBS67cuUKdu/ejZMnT6Jx48YAgEWLFqFLly74/vvvYWNjg9WrVyMzM
xO//vorlEol6tWrh7Nnz2LevHlSwFm4cCE6deqEiRMnAgC+/vprhIWFYfHixVi+fHkp7jERER
G9jnI/xuXAgQOwsLBAnTp1MHr0aDx8+FBaFhkZCZCTMzMym0AICnpycqVKiA48ePS21at24NpVI
ptfHy8kJ0dDQeP34stfH09NTarpepeXFyIjIwutKyMjA8nJyVovVoviIiKlvlOrh06tQJv/32G8LD
w/Htt9/i4MGD6Ny5M3JycgAgAA8fHxsLCw0FpHX18flStXRnx8vNTG0tJKva5C0vSGBgI
DQajfSytbV9vZ0lIiKiV9LppaJX6devn/bGvn/RvFxcXNGjQAI6Ojjhw4ADat2+vw8oQf39/+Pn5Sd
PJyckML0RERGWsXJ9xeVHNmjVRtWpVXHNmjVRtWpVXL9+HQBgZWWFhIQErTbZ2dl49OiRiRNC7GysoK9+7d02q
TN/2qNoWNrQGejb1Rq9VaLyIiIipbsgout2/fxsOHD2FtbQ0A8PDwQGJiIqKioqQ2+/btQ25u
Ltzd3aU2ERERyMrKktqEhYWhTp06qFSSpktQmPDxca1thYWHw8PAo610iIiKiYtBpcElJSs
2dx9uxZAEBsbCzOnj2LuLg4pKSkYOLEiTh27Bhu3Bhu3LiB8PBwdO/eHbVq1YKXlxcAwMnJCZ06dc
KIESNw4sQJHDlyBD4+PujXrx9sbGwAAB9//DGUSiWGDRuGS5cuYf369569Vi4cKHWZZ5x48Zh9+7
dmDt3Lq5evYqAgCCCcOnUKPj4+b/yYEBERUeFKFFzatWuWHxMTEfPOTk5PRrl27vu27Ivdz6tQpvPvu
u3j33XcBAH5+fnj33fp06Onp4fz58/jgJw8fv/jggw/wzjvvvYNiwYWjUqBEOHToxEl/Uo19bF69WrUr
VsX7du3R5cuXdCyZUUutZ7RoNBqEhoYiNjYWjRo1s/8Fy//gX5hzzOgZZv75dnEWVu4NGF2+8wD19lEmMPadmIC8/Qx/xK6oQCxKvqnPr+CGJqXy4Fs7/Exbxe6Vo/qnFPzwUn7pnLVWnFaD9/XuBnbpQPLuWM8/iJdoRFC4vNXS10b4pzEsB/TUZHBbZxHGbaKQYxUhqJnjpG8UIzTIrjhExVHVH5ZVcVbDWiGN9LZAhxWa
M2aNXHixAmYm5tLy5RKJSwoLKCnp1fqKqRIREREBxQwudnZ2AIDc3NwyKYaIiIjoZUr85Nxr16
5h//79Xy2Ew3VW8bRIRBkXaLuZ2RuV8I4JXTcKpLiLtMhZ7GiOOe0lMiIIiqIXlSi4nDlzRmu16
SBzFzs2djzpw5mDx5cmnXnXQ0RERFSoEj2A7vHjx4ZqQJ6ahS/9mOWKz+jdu3dp1JERET0UiKLr1790ZoaGhp10JERET6Ej2EEkSTiYmOGhFRvqqFUPlEq3DqLlGjp10JE
RET0UiW6VFSrVi1MmzYNNx44dg4LYDvgsuhfbnkrFsvRBlFWJbuLyp8YWJiF1sw2u9
PCg1jKFQsHgQkRERGWiRMElNja2tOsgIiIeqUSjXEhIi0oUSnXH59NNPPTX7r8119/LVExRE
RERC9TouDy+PfjremsrCxcvHgRiYmJBXX75IhEREVFpKFFw2b2cm5cPD7n0A5dEdHVtBNNHDV+7KCI
iIqKClNoYlwoVKsLRWZGnMCnKI8KzSDPzw/s588vrS6xriIiIiJTdq4NdHGIRmBu6QoUKmD3DLiXvS4Ii6pRIFIzzz3799HdHQ0AKBOlhCl
7t27CAkJweDBg0ulMCII1qIXlSi4nDlzRmu6QoUKMDc3x9x7/rknteq5xERERERFRSZUouOzfv7
yAiIiJ6pRIFlzzz3799HdHQ0AKBOnTowNzcvlaKIiiIiClKiwbnmpan49NNPYW1tjdatW6N169awsbHBsGHDkEpKSnHTdWl1tjdatW6N169
awsbHBsGHDkJaWVto1EhEREQoYXDx8/PPDwMHsJiiIxMRHbh3DwYMHw9/PHjRs3cHXZeWkdLNTpZq9sHX9G9q+LrndYn7sjSZM6dsb6FanbhXk9ek2WFPMWFgiIvWhh
IiAhACS8Vbd68GZs2bULbm2leV26dIGhoSkotcSbQIGNvohoSH69OmSDcmWvVZ9REKerRJISnXFJS0uDpaVlvvkW
Fha8VERERERlpkTBxcPDAzNmzEB6ero7+nTp5g5cyY8PDxKrTgiIiKi55XoUtGCBQvQqVMnV
K9eHa6urgCAc+fOQaVSYSIsYtpToDi4uKCa9euYfXq1bh69Soh/RG2aeVoawDU
1LtUAiIiKiPCUKLoGBgbC0tMSEISESO05v/666+4+4f/8+Jk+eXCrFERERET2vRGNcxvR9StWzf
f/Hr16mHX1RmhWvwRXRQRERFRQUoUXOLj42FtbZ1vvrm5nbD5vFvvm5Oe7evfvaRRRRREREVpETBxdbWFkeOHMk3
/8iRI7IxgcTmThGKqkrUUKqxMIdMiI5IgQUHpGLLNGjICvry+ysrLQr107AEB4eDhr0+4eaxDgmAFyCtOEFpuuSIMR
SIePnyMD7//HJmZmQAAAwMDTJ48N9Gf4+/qVaIBEREVGeEgUQxMKBb7/9FtOmcTcOVK1dgaGiI
VrQ6VSlXZ9RERERJISBZc8JiYmaNKkSWsNRKRWnVkRERERPRSJRqcS0RERKLDC5EREQkwuukWERERJBbsMkRERERJBsMLkREREJBsMkRERERJBsMkRERERJBsMLkR
MLkRERCQbDC5EREQkGwwuRERERJBsMLkRERCQbDC5EREQkGwwuRERERJBsMLkRERCQbOg0uERER
eP/992FjYwOFQoGtW7dqLRdCYPr06bC2toahoSE8TP1x7do1rTaPHj3CgAEDoFarYWZmXYWZmhmHDh
iElJUWrzfnz59GqVsYGBjA1tYWQWFBv1cwEWnr9dSm3Cq38pDE7i4uGDznzp2lvr9ERET0enQaXaXF
JTU+Hq6oolS5YUuDwoKAg//vgjli9fjuPHj8PY2BheXl5IT09LNZnvuKvD4+8V/lGdc3lgDMCCXLl1CWFgyduzYgYiICIwcOVJanpycjI4dO8LZ2RlRUVH47rvvMGPGDPz0009Smy NPc9Sx1n0KNHDwDPrzbY2NhgwoMACXL1CWFgYduzYgYi
ICIwcOVJanpycjI4dO8LOzg5RUVH47rvvEBAQgJ9++klqc/ToUfx/Dhg3DmTNn0KNHD/To
0QMXL14su50nIiKiYlMIIYSuiwCefeP0li1b0KNHDwDPzrbY2NhgwoQJ+OKLLwAASULJsLS0R
HBwMPr164crV67A79m3Y2Nhg2bJl+OqrrxAfHw+lUgkAmD

```
JlCrZu3YqrV68CAPr27YvU1FTs2LFDqqdZs2Zo2LAhli9fXqT6k5OTodFokJSUBLVaXVqHRTJ
h12+l3idReTO38yBdl1BiCcsm6boEojJnMTr/FYvSUJzfoeV2jEtsbCzi4+Ph6ekpzdNoNHB3
d0dkZCQAIDIyEmZmZlJoAQBPT09UqFABx48fl9q0bt1aCi0A4OXlhejoaDx+/Fhq8/x28trkb
acgGRkZSE5O1noRERFR2Sq3wSU+Ph4AYGlpqTXf0tJSWhYfHw8LCwut5fr6+qhcubJWm4L6eH
4bhbXJW16QwMBAaDQa6WVra1vcXSQiIqJiKrfBpbzz9/dHUlKS9Lp165auSyIiInrrldvgYmV
lBQC4d++e1vx79+5Jy6ysrJCQkKC1PDs7G48ePdJqU1Afz2+jsDZ5ywuiUqmgVqu1XkRERFS2
ym1wcXBwgJWVFcLDw6V5ycnJOH78ODw8PAAAHh4eSExMRFRUlNRm3759yM3Nhbu7u9QmIiICW
VlZUpuwsDDUqVMHlSpVkto8v528NnnbISIiovJBp8ElJSUFZ8+exdmzZwE8G5B79uxZxMXFQa
FQwNfXF7Nnz8Zff/2FCxcuYNCgQbCxsZHuPHJyckKnTp0wYsQInDhxAkeOHIGPjw/69esHGxs
bAMDHH38MpVKJYcOG4dKlS1i/fj0WLlwIPz8/qY5x48Zh9+7dmDt3Lq5evYqAgACcOnUKPj4+
b/qQEBER0Uvo63Ljp06dwnvvvSdN54WJwYMHIzg4GJMmTUJqaipGjhyJxMREtGzZErt374aBg
YG0zurVq+Hj44P27dujQoUK6NmzJ3744QdpuUajQWhoKLy9vdGoUSNUrVoV06dP13rWS/Pmzb
FmzRpMnToVX375JWrXro2tW7eifv36b+AoEBERUVGVm+e4yB2f40L0+vgcF6Lyjc9xISIiIio
GBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIi
kg0GFyIiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciI
iKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg0GFy
IiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg0GFyIiIpINBhciIiKSDQY
XIiIikg0GFyIiIpINBhciIiKSDQYXIiIikg19XRfwvyYnJwdZWVnFXk+tpyyDaqgwAsDT3Gxk
i1xdl0JERM9hcHlDhBCIj49HYmJiidbvUNWxdAuilxIAcnJzcT31IS6lPdB1OURE9B8MLm9IX
mixsLCAkZERFApFsdY3ePK4jCqjAgmBrIxMKB88+4gwvBARlQ8MLm9ATk6OFFqqVKlSoj70My
qWclX0KvoqJSoDqJWTjeinj3jZiIioHODg3Dcgb0yLkZGRjiuh4qqoUkKvQgUYVmDGJyIqDxh
c3qDiXh6ickChgAIA3zkiovKBwYWIiIhkg8GFSiTy0BHUMDNHUmKSrkshIqL/IQwuMvfwwQN8
6TcRzeo3RC2Lamj0jjM++ag3Th47Xmrb6NO1OwKmfKU1r5F7E5yKvgi1Rl1q2ykpv9E+GP7xI
F2XQUREbwBHHMrcZwOHIisrC/OWLkYNezs8uH8fhw9G4PGjsr19WqlUwsLSsky3QURE9CKecZ
GxpMQknIg8Bv+AaWjeuiWq17BFw0Zu8PHzRccunaQ2k8b4oqFjXTjbOqDf+x/i8oWLUh/zAoP
QqWVbbF63Ac1d3FCvRk14fzoCKU9SADw7m3HsyFH8uvwn1DAzRw0zc9y6GZfvUtHG1WtRv4Yj
9u4ORdvGzfCOdQ18NmgonqalYeOadWju4ob6drUwfZI/cnJypO1nZGRg9tQZaOLkgjo2dvigv
RciDx2Rluf1ezB8H9o1bY661ewwsGcf3IuPl+rftHY9Qnfukup7fn0iInq7MLjImLGJMYxNjL
EnZBcyMjIKbDN6yDA8uP8AqzatQ8iBvajv2gD9u/dE4uP/npG5eeMGQkN2YuX61fh13WocO3I
USxcsBAAE/N83aNS0CfoPHohT0RdxKvoibKpXK3BbT58+xcoff8biFT/h903rcOzwUYz4ZDD2
h+1F8Ma1WPDjUqwJ/g0h2/6S1pk2cQqiTp7E4hU/Yc+RA+ja4wMM6tUXsTExWv3+uGgp5v+4F
BtD/sK/t//FnGkBAIDPxnyObh92R1vPdlJ9jdybvO6hJSKicqpcB5eAgAAoFAqtV926daXl6e
np8Pb2RpUqVWBiYoKePXvi3r17Wn3ExcWha9euMDIygoWFBSZOnIjs7GytNgcOHICbmxtUKhV
q1aqF4ODgN7F7r01fXx9zlyzCprXrUd+uFj706oJvZ83GlYuXAAAnIo/h3OnTWLZqBVzfbQgH
R0dMnT0Tao0aIdu2S/3k5grMXboYdZyd4N7cAx/17YMjBw8BANQaNSpWrAhDQ0NYWFrCwtISe
np6BdaTlZWFOfOCUN+1AdxbNEeX7u/j5LET+G7RArxTtw48O3WER6sW0hmRf2/dxsbVa7Es+F
e4N/eAvYMDPhvjjcbN3LFh9Vqtfr+Z/x1c320Il4auGDJiGI4cjAAAGJuYwMDAAEqlSqpPqeT
3OhERva3K/RiXevXqYe/evdK0vv5/Sx4/fjxCQkKwceNGaDQa+Pj44KOPPsKRBr
166wsrLC0aNHcffuXQwaNAgVK1bEN998wAwCIjY1F165dMWrUKKxevRrh4eEYPnw4rK2t4eXl9
WZ3tgS6dH8f7bw64ETkMZw5GYUDe8OxfOFiBP0wH2lpaUhNSYVrzXe01kl/mo6bsTek6eo1bG
FiaiJNW1pa4sH94j/i3tDICPYODtJ0VXNzVK9hC2MTE615D//T99XLl5GTk4O2jd21+snMyES
lypUK7deihPUREZH8lfvgoq+vDysrq3zzk5KSsGLFCqxZswbt2rUDDAKxcuRJOTk44uduwYmjVr
htDQUFy+fBl79+6FpaUlGjZsiK//hqTJ09GQEAAlEolj+yvWlIgC4du0abGx
sULNmTQwYMABxcXEAgKioKGRlZcHT01NqW7duXdSoUQORkZEAgMjISLi4uMDyubtfvLy8kJyc
jEuXLkltnu8jr01eH4XJyMhAcnKy1qu8qF2nDtLS0lDftQHu30uAnp4+7GvW1HpVLsZ3JlVUK
pH73IDa0lK/QQPk5OTgwf37+eorzh1LFZVKrQG/RET09irXwcXd3R3BwcHYvXs3lil1bhtjYWL
Rq1QpPnjxBfHw8lEolzMzMtNaxtLRE/H/uOImPj9cKLXnL85a9rE1ycjKePn1aE1ycjKePn1aaG2BgYHQaDT
Sy9bW9nV3t9geP3qEfu9/iD/Xb8SVi5cQd+MmmdzdhuU/LELHLp3Qqm0buDVtjBEDBiFi337c
uhmHU8dPIOjrOTh35myRt1O9hi3ORJ3GrZtxePTwoXTG5HXVrOWID/v0gt8oH+z6awfibtzE2
ajTWDxvAcL3hBa5H9satrh66TJirl3Ho4cPpe+GIiKit0+5vlTUuXNn6d8NGjSAgAAMDgxLt960kDkbVhezMLNy9dRthD2KQnJOp63LeenM7y/cJyAnLJum6BKIyZzE6qEz6fdnv
```

0BeV60tFL0pJSUFMTAysra3RqFEjVKxYEeHh4dLy6OhoxMXFwcPDAwDg4eGBCxcuICEhQWoTF
hYGtVoNZ2dnqc3zfeS1yeuDiIiIyo9yHVy++OILHDx4EDdu3MDRo0fx4YcfQk9PD/3794dGo8
GwYcPg5+eH/fv3IyoqCkOHDoWHhweaNWsGAOjYsSOcnZ0xcOBAnDt3Dnv27MHUqVPh7e0NlUo
FABg1ahT++ecfTJo0CVevXsXSpUuxYcMGjB8/Xpe7TkRERAUo12Ncbt++jf79++Phw4cwNzdH
y5YtcezYMWl8xvz581GhQgX07NkTGRkZ8PLywtKlS6X19fT0sGPHDowePRoeHh4wNjbG4MGDM
WvWLKmNg4MDQkJCMH78eCxcuBDVq1fHL7/8IptboYmIiP6XlOvgsm7dupcuNzAwwJIlS7BkyZ
JC29jZ2WHnzp0v7adt27Y4c+ZMiWokIiKiN6dcXyoiIiIieh6DCxERECkGgwsRERHJBoMLERE
RyQaDCxEREckGgwvpTOShI6hhZo6kxJc/obi5y7OvNSAiIirXt0P/L5iw67c3ti3f5l3e2LaK
opF7E5yKvgi15tnjnTeuXouZ/lNxMS5Gq932/aEwMjLSRYlERFTOMLiQziiVSli88M3cBalSt
eobqIaIiOSAl4ropfp07Y5pEydj2sTJqFejJlxr1sH3swOR992ciYmJ8P3MG/XtauEd6xoY1K
svYmP+e8bkdtwtDO07APXtaqGOjR3aN2uJfaFhALQvFUUeOoIJ3mORnJyMGmbmqGFmjnmBz77
M6/lLRWOGf4bPhw7XqjErKwuuNetg09r1AIDc3FwsnrcALRo0Qm0rW3i1aIuQbX+V+bEiIqKy
xzMu9Eqb1q5H308G4K/wUJw/cxZTfCfAxrY6Ph48EBNGj0HsP/9gxdrfYWpqisCAWRjcuz/Cj
x9BxYoVMXXiZGRlZmHTzr9gaGyEa1f/hrGxcb5tNHJvghmBszEv8FvsPxkJAAW269G7J0YPGY
7UlBQYm5gAAA6G78fTp0/RqVtXAMCSeQuwZcMmfDP/O9g71sSJI5HwHfk5qlSpgmYtW5ThkSI
iorLG4EKvZF2tGmYEzoZCoYBj7Vq4evkyflm6HB4tmyNs1278uScEjd2bAgB++Hk53Os1xJ6Q
nejWozvu3PoXnT/ohrr1nn0bt529fYHbUCqVUKvVUEDx0stHbdq3g5GREXbv2Ime/foAALZt2
owOnb1gYmqCjIwMLJ63EGu2bkKjpk2kbZ48dhyrg39jcCEikjleKqJXcmvcCAqFQppu1KQJbs
T8g2tX/4a+vj7ebdxIWlapcmU41nLE9ehrAICho4Zj0ffz8KFXF8z95ltcuXjptWrR19dHtw8
/wNaNmwAAaampCN25Gz169wIA3PgnFk/T0jDgw16oW81Oem1etwE3Y2+81raJiEj3eMaFylT/
QQPRpl07hIeG4dC+A1g6fyGmzp6JoZ+NKHGfPXr3Qp+u3fHg/n0c2n8QBgYGaOvZDsCzIAMAw
evXwMrGWms9pVJV8h0hIqJygWdc6JXORJ3Wmj596hTsHWuidt13kJ2djTOnoqRljx89Qsz1GN
Su+440z6Z6NQz8dAh++iMYI3xGY+2qPwrcTkWlEjm5Oa+sp7F7U1hXq4btf27F1o2b0LXHB6h
YsSIAoHadOlCpVPj39r+wr1lT62VTvVpJdp+IiMoRnnGhV7pz+zZmfTkNA4YOxsVz5xH80y+Y
OnsWHBwd0bFLZ0we54fA+d/DxMQE/zfza1hZW6Fjl84AgIApX+G9Du3h4OiIpMRERB46glp1a
he4neo1bJGakorDByPgXL8eDA0NYVjI81t69P4If6xchdjrMVi3fYs038TUBCPHfI5ZX05Dbm
4umni440lSMk4dPwETU1P0/rhf6R8gIiJ6YxhcdGxu50FFancr6UEZV1K4nv36ID09HR+074g
KFfTw6aiRGDDkWd3fL/0BAZO/wqd9ByAzKwvuzZth1ca10hmQ3JwcTP1iMuLv3IWJqSnatG+H
GYFfF7idxu5N8cmnQ+A9dAQeP3oE38kT4ec/qcC2PXr3wqLv56O6rS2aNHPXWvbFV/6oXKUKl
s5fiLhxN6HWaFDf1QU+fr6ld1CIiEgnFCLvgRz0WpKTk6HRaJCUlAS1Wq21LD09HbGxsXBwcI
CBgUGJ+tdVcOnTtTucXeoj4P/m6GT7upadmYW7t24j7EEMknMydV3OW6+oQb48SlhWcMgmept
YjA4qk35f9jv0RRzjQkRERLLB4EJERESywTEu9FIbQrbpugQiIiIJz7gQERGRbDC4EBERGRwww
uBAREZFsMLiQbM0LDEKnlm11XQYREb1BfI6LjhX1MeGqU
thWxsfyfSR5DTNz/PzHKnh16yLN+2zM5xj62XAdVkVERG8agwvJlrGJCYx1XQQREb1RvFREL9
Wna3dMn+SPOdNnwsW+Nhq944x5gf/9kq2kxCRMGuOLho514WzrgH7vf4jLFy5q9fHDd3Pxbi0
nOFW3x6QxvggMmKV1iefc6TP4uEcvuNasg3o1aqJ3lw9w4ew5aXlzFzcAwIhPBqOGmbk0/fyl
ooh9+1HbsjqSEpO0tj1j8pfo9/6H0vSJyGPo2bkbalvZwr2eK6ZP8kdaamqpHCsiIip7DC70S
pvXrYeRkRH+Ct8N/5kzsDDoe0TsPwAAGD1kGB7cf4BVm9Yh5MBe1HdtgP7deyLx8WMAwJYNm7
Bo7gL4B0xDyIFw2FSvjj9+DdbqP+VJCnr174vNu3dga9huODjWxJA+/ZHyJAUAsH1/KABg7pI
fcCr6ojT9vBZtWkOtUWPXX9uleTk5OdixZSt69O4JALgRG4tBvfqi8/vdEHrkAJb8+jNOHTuO
aROnlPYhIyKiMsLgQq9Ut54zxk+ZCAdHR/Tq3xcN3m2IIwcjcCLyGM6dPo1lq1bA9d2GcHB0x
NTZM6HWqBGy7VmACP7pF/Qb+DH6fPIxatZyhO/kL1DH2Umr/xZtWuGjvr1R653aqF3nHfzfwn
l4+vQpjh05CgCoUrUqAECt0cDC0lKafp6enh7e/+hDbN30pzTvyMEIJCclo/MH7wMAls5biB6
9e2H456Pg4OiIxu5NEfDtN9i8bgPS09PL5NgREVHp4hgXeiWnes5a0xaWlnh4/wGuXLyE1JRU
uNZ8R2t5+tN03Iy9AQCIuXYdA4cN1Vre0M0NRyMOSdP3ExLw3exAHDt8BA8fPEBOTg6epj3Fn
du3i1Xnh316obtnJ8TfjYeVtRW2bNiEdh09oTHTAAAuX7yEq5cuY+vGTdI6QgC5ubm4dTMOte
u8U1jXRERUTjC40Cvp61fUmlYoFMjNzUVqaiosrCyxYcfWfOuoNZoi9+832gePHz1GwP/NQTV
bW6hUSvTo0AWZmZnFqtPV7V3YOdhj++YtGDhsCPaE7MTcJYuk5Wmpqfh4yCB8OmpEvnVtqlcv
1raIiEg3GFyoxOq7NsD9ewnQ09OHrV2NATs41q6F86fPoTxEsdCaPuzTExfP
ncfi7xegS/f3oVL99wk4o8eNQdSJk5g2cTIunb+A2JgYhIbswrSJk0v1uBARUdnhGRcdsxgd9
OpGAG4lPSjjSopPoVBg1YZ1CPp6DiZ4j8WjBw9hbmkB9+YeMDc3B/Bs3EncjZuYMy0AGRnp6N
ajO3r174dzp/971iVo0QJM8Z2ALm3aw6aaDSZN/wpzpgZobWvq7Fn4+qtpWLvqd1hZW+PohdM
F1mRfsyYaNnLD2ajTmBE4W2uZU/162BiyDUFff4NeXd6HEAJ29g54/6PupXtgiIiozCiEEELO
RbwNkpOTg6epj3Fn/Bs3EncjZuYMy0AGRnp6NajO3r174dzp/971iVo0QJM8Z2ALm3aw6aaDSZN/wpzpgZobWvq7Fn4+qtpWLvqd1hZW+PohdM
F1mRfsyYaNnLD2ajTmBE4W2uZU/162BiyDUFff4NeXd6HEAJ29g54/6PupXtgiIiozCiEEELX
RbwNkpOTodFokJSUBLVarbUsPT0dsbGxcHBwcICBgUGJ+tdVcOnTtTucXeoj4P/m6GT7upadmYW7t24j7EEMknMydV3OW6+oQb48SlhWcMgmept
YjA4qk35f9jv0RRzjQkRERLLB4EJERERwww
uBAREZFsMLiQbM0LDEKnlm11XQYREb1BfI6LjhX1MeGqU
thWxsfyfSR5DTNz/PzHKnh16yLN+2zM5xj62XAdVkVERG8agwvJlrGJCYx1XQQREb1RvFREL9
Wna3dMn+SPOdNnwsW+Nhq944x5gf/9kq2kxCRMGuOLho514WzrgH7vf4jLFy5q9fHDd3Pxbi0
nOFW3x6QxvggMmKV1iefc6TP4uEcvuNasg3o1aqJ3lw9w4ew5aXlzFzcAwIhPBqOGmbk0/fyl
ooh9+1HbsjqSEpO0tj1j8pfo9/6H0vSJyGPo2bkbalvZwr2eK6ZP8kdaamqpHCsiIip7DC70S
pvXrYeRkRH+Ct8N/5kzsDDoe0TsPwAAGD1kGB7cf4BVm9Yh5MBe1HdtgP7deyLx8WMAwJYNm7
Bo7gL4B0xDyIFw2FSvjj9+DdbqP+VJCnr174vNu3dga9huODjWxJA+/ZHyJAUAsH1/KABg7pI
fcCr6ojT9vBZtWkOtUWPXX9uleTk5OdixZSt69O4JALgRG4tBvfqi8/vdEHrkAJb8+jNOHTuO
aROnlPYhIyKiMsLgQq9Ut54zxk+ZCAdHR/Tq3xcN3m2IIwcjcCLyGM6dPo1lq1bA9d2GcHB0x
NTZM6HWqBGy7VmACP7pF/Qb+DH6fPIxatZyhO/kL1DH2Umr/xZtWuGjvr1R653aqF3nHfzfwn
l4+vQpjh05CgCoUrUqAECt0cDC0lKafp6enh7e/+hDbN30pzTvyMEIJCclo/MH7wMAls5biB6
9e2H456Pg4OiIxu5NEfDtN9i8bgPS09PL5NgREVHp4hgXeiWnes5a0xaWlnh4/wGuXLyE1JRU
uNZ8R2t5+tN03Iy9AQCIuXYdA4cN1Vre0M0NRyMOSdP3ExLw3exAHDt8BA8fPEBOTg6epj3Fn
du3i1Xnh316obtnJ8TfjYeVtRW2bNiEdh09oTHTAAAuX7yEq5cuY+vGTdI6QgC5ubm4dTMOte
u8U1jXRERUTjC40Cvp61fUmlYoFMjNzUVqaiosrCyxYcfWfOuoNZoi9+832gePHz1GwP/NQTV
bW6hUSvTo0AWZmZnFqtPV7V3YOdhj++YtGDhsCPaE7MTcJYuk5Wmpqfh4yCB8OmpEvnVtqlcv
1raIiEg3GFyoxOq7NsD9ewnQ09OHrV2NATs41q6F86fPoTxEsdCaPuzTExfP

8HPT09bNv0Jw4fOIjVWze9emUiIqIXMLhQmVIoFNgXtheL5s5HRkYGHGs54sffVqJV2za6Lo2
IiGSIwYXKlIGhIdZu26zrMoiI6C3Bu4qIiIhINhhc3iCOg5YhISAA8J0jIiofGFzegIoVnz15
Ni0tTceVUHFlZWQiJzcXT3OzdV0KERGBY1zeCD09PZiZmSEhIQEAYGRkpPUAtqLIzswqi9KoM
EIgKyMTjx48xPXUh8gWubquiIiIwODyxlhZWQGAFF6K6/HTlNIsh15BAMjJzcX11Ie4lPb2PE
OHiEjuGFzeEIVCAWtra1hYWCArq/hnTzZEbC39oqhQAsDT3GyeaSEiKmcYXN4wPT096OnpFXs
9PrWViIiIg3PzWbJkCezt7WFgYAB3d3ecOHFC1yURERHRfzC4PGf9+vXw8/PDjBkzcPr0abi6
usLLy6vE41KIiIiodDG4PGfevHkYMWIEhg4dCmdnZyxfvhxGRkb49ddfdV0aERERgWNcJJmZm
YiKioK/v780r0KFCvD09ERkZGS+9hkZGcjIyJCmk5KSADz7hsuykJH2tEz6JSpPyurz8yY8eZ
rx6kZEMmdQRp/RvM9+UR7UyuDyHw8ePEBOTg4sLS215ltaWuLq1av52gcGBmLmzJn55tva2pZ
ZjURvuyUYpesSiOhlJvxQpt0/efIEGo3mpW0YXErI398ffn5+0nRubi4ePXqEKlWqFPvhclT+
JCcnw9bWFrdu3YJardZ1OUT0An5G3y5CCDx58gQ2NjavbMvg8h9Vq1aFnp4e7t27pzX/3r170
sPjnqdSqaBSqbTmmZmZlWWJpANqtZr/KRKVY/yMvj1edaYlDwfn/odSqUSjRo0QHh4uzcvNzU
V4eDg8PDx0WBkRERHl4RmX5/j5+WHw4MFo3LgxmjZtigULFiA1NRVDhw7VdWlEREQEBhctffv
2xf379zF9+nTEx8ejYcOG2L17d74Bu/T2U6lUmDFjRr7+9LIYpy7xERERFROcAx
LkRERCQbDC5EREQkGwwuREREJBsMLkTPOXDgABQKBRITE1/azt7eHgsWLHgjNRHR6wkICEDDh
g11XQaVEg7OJXpOZmYmHj16hBgYUCgQHB8PX1zdfkLl//z6MjY1hZGSkm0ICEXMTXZdBpYS
Xikh22rZtCx8fH/j4+ECj0aBq1aqYNm2a9K2ijx8/xBBg1CpUiUYGRmhc+fOuHbtmrT+zZs3
8f7776NSpUowNjZGvXr1sHPgDal4oOHDiAoUOHIikpCQqFAgqFAgqFAgEBAQC0LxV9/PHH6Nu3r
1aNWVlZqFq1Kn777TcAz57CHBgYCGxaGsLV1RWbNm0q4y4NF9Oa0bdsWY8eOxaRJk1C5cm
VYWVlJnxcASExMxPDhw2Fubg61Wo127drdrh3LlzWn3Mnj0bFhYWMDU1xfDhwzFlyhStSznT55
Ehw4dULVqVWg0GrRp0wanT5+Wltvb2wMAPvzwQygUCmn6+UtFoaGhMDAwyHcWddy4cWjXrp00
ffjwYbRq1QqGhoawbtXF2LFjkZqa+trHi4V4fgwvJ0qpVq6Cvr48TJO5g4cKFmDdvHn755RcAw
JAhQ3Dq1Cn89ddfiIyMhBACXbp0QVZWFgDA29sbGRkZiIiIwIULFzF/Dtt98W+NdY8+bNsWDBAq
jVaty9exd3797FF198ka/dgAEDsH37dzuzZg7S0NHz44YcAnn2b+G+//Ybly5f0qV
LGD9+PD755BMcPHiwLA4PkU6sWrUKxsbGOH78OIKCgjBr1iyEhYUBAHr37o2EhATs2rULUVFR
cHNzQ/v27fHo0/v27fHo0SMAwOrVqzFnzhzzhx8x8++23iIqKQo0aNbT/p88eYLBgwfj8OHD
OHJXbrgryZMnAJ4Fk69K17t69K00/r3379jAzM8Pmz585zZuleTk5OVi/0fj0GDBgaAAIIiiYn
3Qs2dPnD9/HuvvfvX8+fhw4fh4fh4+NT+geNik8QyUybNm2EkNzM3NleZNnjxib//////lsAEEe
OHJGWPXjwBgaGooNGzYIIYRwcXEREydOLPLoyhbhUajydfOzs5OzJ8J8/
XwghRFZWlqhatar4+cxFUIkZ6eLoyMjMjMjMTRo0e1+hg2bJjo379/sfefqDxq0
6aNaNmypda8Ju7++0+PtPyZ9mypda8Jk2iMmTJ4tDhw4d+8tZa3qJFC+Hq6ir7
roNnNycoSpqanYvn27NA+A2LGTNmaPUzkyY7Ojq0a5dOZ549QqVSSZ/7ocOGiTm32PdWJ+2zLr
1cejQIVGhQgXVmjVrBoVCIU17eHjg7eeHu7i4tq1KlCrU
qYMrV64AAMzZ2aG1ACA1NRXbtm2T/nq/060
tLS0KFDB+lau4mJCX777TEx81raJypMGDTRoTVtbWyMhIQFjIQHnzp1DSkoKqlSpAvVKqlpov
4D0dHRaNq0qdb6L07fu3cPw2mpataXaKBWq5GS
ocOGiTRycS/9zhg8fDi8vL2TAD0NBSBgYG
O3cuxowZU+I+BwwYgDZt2iAhIQGhoaEwNDSEl5cXAGmXRjn9p8e6MOyvlhyv+uHPdEhDAGAFhuKCbzs4NKpYKHhwcyMzOLVWeTJk3g
QoFcnNzkZKSAmtraxaw4cCDf0nlhoSgGDx6Mhw8fyYHChbcs4NKpYKHhwcyMzOLVWeTJk3g6O
iIdevWYfTo0diyZgQuJ5NuifwQrrJ0/Phxrem8693
Izs7G8ePH0bx5cwDAw4cPcePHDdWdnZ6m9ra0tRo0ahvGsHf3x8/x8///xzgcFFqVQiJyfnlfU0
b94ctra2WL9+PXbt3oX1ffX/sdfqDxq0
erg5MmTGDRokDTvxTEqR44cwdKlS9GlSxcAwAwk1bt/DgwQOtNhUrVizSZ3bAgAFYvXo1qlevjg
oVKqBr165a9V6fBm1atUq6i7SG8RLRSRLcXFx8PPzQ3R0NNauXYtFixbh888/h
RI0bg8OHDOHfuHD755BNUq1YN3bt3BwD4+vpiz549uHHjBk6fPo39+/fDycmpwHXb29sjJSUF4+
PDwcDx48QFpaWqE1ffxx+lbfzxx1i+fDnCwsK1i+fDnCwsKky0QAYGpqii+++ALjx4/H7t27c
WpV6R4YonLI09MTH4e6NGjB0JDQDQ3HjxgwcPcPXX331fU6dOgbHBK5P0UwTTY0ph2rVrmD
17Ns6fP691Sbh27dr4/fffceTIEXz44Yf4e+YHfsPH78uNCaBgwYgN
nT2POnDno1auX1tnPyZMn4+jRo/D8cHZs2dx7do1bNu2TbPKP0kCtb4enTp2jatCm8
vb0xbtw4jBw5EsCzowaNWqEbt26wU11UsMXHrOV+ZMsY+6ejNasA0rTNP6/hvUKAzQBM0bQtkB4PVG
gK307x5c4waNQp9+/aFubk5goKCS2wAAcBHuHsEHJTrRm15vbzgy2iR5ZasSyR4yxRMGntVZ9XXpPVvzvsb
mBg4NDKR0RovJLoVBg586daN26NYYOHYp33nkH/fr1W82bN2zwxR30Z3cBA
bG4shQ4bAwMBA6mfFihV4/Pgx3NzcMHDgQIwdZa25o7dy7CsCsJ2uLdKgqIH5999t9CaatWq
haZNm+L8f+Naf2gAz8bqHDx4EH///w/TdatWqFd999/F9OnT4eNjU0pT0pHhUqKT45l2Snbdu2aNiwI
R+5T/SW69ChA6ysrPD777/HoQw3z7Lf+H15cX9PT0sHbtWuzdu1dgxRHg
YXIiLSubzLSXPmzEF6ejrq1KmDzDzZzs3w9PTU9elUTnDS0VEREQkGxycS0RERLLB4EJEREESyweB
CRERESsHgQkRERLLB4EJEREQkGxRG8le3t7PqSQ6C3E4EJEshYCHFzgNwyfPHlS+hoIXTpw

4AAUCgUSExN1XQrRW4EPoCOit5K5ubmuSyCiMsAzLkRU5jZt2gQXFxcYGhqiSpUq8PT0RGpqK
gDgl19+gZOTEwwMDFC3bl2tL7y8ceMGFAoF/vzzT7z33nswMjKCq6srIiMjATw7mzF06FAkJS
VBoVBAoVAgICAAQP5LRQqFAj/++CO6desGIyMjODk5ITIyEtevX0fbtm1hbGyM5s2bIyYmRqv
2bdu2wc3NDQYGBqhZsyZmzpyJ7OxsrX5/+eUXfPjhhzAyMkLt2rXx119/SfW/9957AIBKlSpB
oVBgyJAhpX14if63CCKiMnTnzh2hr68v5s2bJ2JjY8X58+fFkiVLxJMnT8Qff/whrK2/whrK2txebNm
8U///wjNm/eLCpXriyCg4OFEELExsYKAKJu3bpix44dIjo6WvTq1UvY2dmJrKwkskZGRIRYsWC
DUarW4e/euuHv3rnjy5IkQQgg7Ozsxf/58qQ4Aolq1amL9+vUiOjpa9OjRQ9jb24t27dqJ3bt
3i8uXL4tmzZqJTp06SetEREQtVotgoODRUxMjAgNDRX29vYiICBAq9/q1auLNWvWiGvXromx
Y8cKExMT8fDhQ5GdnS02b94sAIjo6Ghx9+5dkZiY+GYOPNFbisGFiMpUVFSUACBu3LiRb5mjo
6NYs2aN1ryvv/5aehHh4CCH+G1x++eUXafmlS5cEAHHlyhUhhBArV64Go0mX98FBZepU6dK05
GRkQKAWLFihTRv7dq1wsDAQJpu3769+Oabb7T6/f3334W1W1tXWh/aakpAgAYteuXUIIIfbv3y8
AiMePH+erkYiKj2NciKhMubq6okOPHeHQGq3tbU1ACAhIQm3EKFiFXFx0ZqXnp6O5ORKqNVqnDt3DkeO
MGcOXOkNjk5OUhPT0daWhqMjIzy9WRhQMj0ZpQ0Qhb1hKMzmkNRW2TXUmkNeBpUsUEDhBAjQhZgboZNE
OxaNEifPXVV9i+fTsA4Oeff4a7u3u+dZ5XXsWJF6d8d8KhQIAkJubW+xciPzvmZvUbIgJtOSu5
jj/L1ZWBgUGC/ef2UpD4iejUGFyIqcwqFAi1atcECLF1i0atECLHxv+sc...

VqlWhVquLPO+PP/4IHx8fqNVqLFiwAE+ePMHatWvRtm1bxMTEoFatWmjbti0UCgUiIiKkRw9H
jx6Fnp4ejh07Jo119+5dXLx4sVRewg0PD0e3bt3g7u6OWbNmQU9PT/old/ToUbRo0UIW369fP
zg6OiIwMBBnz57F999/DysrKyxYsECKGT58OLZv346hQ4eiVatWOHLkCLy9vWXj9O7dG5cuXc
LPP/+MpUuXwtLSEgBQvXp1KebYsWPYuXMnxo0bBxMTE6xYsQJ9+vRBcnIyqlWrBgAYO3Ysfvn
lF4wfPx6NGzfGvXv3cOzYMVy4cAHNmjUrct4nTpwagCJjnj7hvT0dFmbiYlJhXnsOHToUHz5
5Zc4cOAARo0aBQAIDQ3FtWvXMGLECNjY2CAhIQr169HQkICTp48KRXKr7rua9euhZOTEz788
EMYGBjg999/x7hx45CXlwc/Pz8AwJ07d9ClSxUr14dX3zxBczNzXH9+nXs3LlTluOYMWc3oCwzB
yMESNGYMKECUhKSsKqVasQExOD48ePo1KlSli2bBk+++wzVKlSBV999RUAwNraWjaOu7s7du3
aVZaXk94lgugfbuPGjQKAOHjwoLh79664fOm66efOm+OWXX0T16tWFSqUSN2/elGI7deokXFxcxcLNn
z6S2vLw80bp1a1GvXj2pLSAgQFSqVEncv39fasvKyhLm5uZi5MiRBc6dlJQBDi0aNHwtzcX
IwaNUqWY2pqqjAzM5O1Ozk5iX79+hUAxIUF4QQQuzcuVMAEOfOnXvlNfDx8R8
GVK1cusj8vL0/Uq1dPqNVqkZeXJ7U/efJe7s6dO0tts2bNEgBk8xRCiF69eolq1apJ+9HJ+ektnPnzgkAYuXKlVKbmZmZ8PPz
R0QKAmDRpkixu+PDhAoCYNWuW1LZo5SLZdXoRAKFUKsWVKbBknPnzgkAYuXKlVKbmZmZ8PPz
K/oiFGHGjBkCgHj06FGh5y5s27hxoxDi7+vq4OBQ4Lj8a/QiBwcH4ePjI+0fOnRIABCHDh16Z
X75P0NnzpwpMsbMzEy4ublJ+0+ePCkQ8/PPPPwsAIiIiQmp71XUvbAy1Wi1q164t7f/222+vze
3o0aMCgNi8ebOsPSQkEC7k5OTaN++fZFjffvttwKASEtLKzKG/jn4iro/3h5eaF69eqws7P
Dxx9/jMqVK2P37t2oWbM2OjPRbMmAOD+/fsIDw79vIrI6SnpyMXPR337t2DWq3G5cuXXpVX/fv3
R3Z2tuy/VA8cOIHDx++nFRaGgoHj58iIEDB0rjp6enQ19fHy1pyYXL1/GoEDcO/ePSmvx48fo/2PZ
49fzp07h9GjR8PS0lJqP3r0KMzeHs7PxG1yY2NhaXL1/GoEGDcO/ePSmvx48fo1OnToiIiC
iwfHrs2LGyfU9PT9y7dw8aj8Nk8V99tlnJc7Py8ldvqhffxcm/A9WqVSA
zNzfHqVOncPv27RKNfe/ePRgZGcl6x9agiobHvHbryUKVKFdlqqhffxcm/A9WqVSA
eO0ju8LGyMjIQHp6Otq3b49r164hPHp6Otq3b49r164hIyMDAKQXk/fs2YPVq1cx9mxYwfMzQuXNn2c+8u7s7q
lSpIvuZf538u60v31GjfyY+oiL6P6tXr0b9+vWWRkZGBDC4Ouvv8bXX
9d6Bh37tzBe++9B1dXVzRs2BDbtm2Dr68vgL8fvL8fT1laWuL998M999NnhyaozTnoXRwj8lj8LXXBwlpvQ
xbt06XLlyBVevXoVCoYCHh4dU+GYe/Iv4eaNQpHjx5F7ZwtX5FmzhUbWZfGSx1ZqIyHB7ZbA4fxnb28v/v6g
e/DgAUxNTXHjxg0Ye3bt0S5I7Ozp0Kdz6HuHkPAHH+s/dJbbDTgDJGTfSwuYfFh3/
ahdu3aJz/eimjIVravV+ztuUmZkJKysyraf/+/fuYM2cOtm7dijt37shi8gcSamvBvSLHWY2G3F3UMLV2KvJJhxmiN7+/
jIAu9WZWRkwMzMDO3bt0efPn0wZ84cLF26zFQ06dEDPnj0xaNAg6e/W5cuXNkZGRIcvvRS/n9yp
CCAB4K+83UcXHAofo/7Ro0UJahdkGzDZQ0bdsWgwYNQmJiIQpUqSLdoqZg2bVqRb/4X+/v3/n/v37
Y/78+UhPT4eJiQl2l796NgQMHwsCg6Lb2ef48ccfC/0slkOtFPSlRePbeduZdLQAgIiIyIeN7NmzVC5c
mV4enpixYyOyVyMzMMRExMDObPnk1/CK1F0XosWLUPuVkuKcq1+/fvD09M9M
Rvv/2GAwcONiRViwYAF27Z27tyJbt26Ys2YN3rlzB99//32R527btm2bNuHSpEmoXr16kX1bsDXdFtGc+fn4
yMmQ/m/369cfg7++fF7Fz2bSj/fXbt2LdYHGV69ehVCvN40gf1RcJGLnNmdyhm/uNtmWdGuVVIuZSvTPuVW
unSpNIZCocAvv/yCkydP4NzffvvsffXqt/Nfq83L2+2iBkBGd6OsoCzwoanlUpC6pjxyVw1bN6xTVsKzz9K
30h+mdjgUNUCH19fQGBqJjx45YtCDIJB9O6zyfXlqAHIg9JenpGXZd3bABZcNQECvBz9/
QYMGDAK4/Jf8xiZWX12nPY29vD3t4eR48rCGWMB8wlTyxDCz0dTpxWSEotVUHsqixLMAPhvyP8lBrsxPmr3lv0BkAp5Y+pmgyG
yMjU1LbW7FQ4ODsjLy0NSUhLq1asntV+5cqVAbGn9F3mNGjzHyHCnSm6H5FpThEdPMj7JaD/Lt1AOtH6Cm/+HTl7n
VxY4DRs2BPD3aqqSfpZM1apVC/2QoBUlD948ABhYWGM2cOzZs6cKcXfxeuw1bVQKeqVZhLqgOxr
6V7UVHX/ffff0dWVhZ2794tuZpu1tbXn8ZdftM3mtYaXH+IpHbrm4vr22xDDpUKr5F4uz4VTk
x4EG3atHntMvbXnR5ngLkqJGcKtmmThb34+CuLLbxmXx7htJxdoOkA0OIGFOC2DWVVDTcfVnEeUoQTUJM8Vp/EF0jPjuzMrBvDe
/kZKSUiD+5Y+Jb9SoEVxcxcXLBt2zZs27YNYbHNYIKyLhCM+7imfMWnFpOwd8tFGMB5ZU4+z6A
+vRpqcBp2rQpTExMEBQUBMjI7i7u5d06gW4u7ujTp06O6775CZmfnavIoj/xfumjVrZO0O0rV
64sEFu5cmUA0PoTdXNzcws8bVhoYKsrSCJ8ApOrOiSOOKkVHdSMvmOJZ/krSi1CRkqiBUdnkX2zUGz
77rcRjaSM8PBBzz5s2Do6MjBg8eDO/3+U7asmXLChfwvgwYCM58m
/A5h/3fv164fc3FzmzevPlcnJpJGrKxJW2c4ZQgHfNMzpjSJRmX60l/7+IeAeH6BX8/f3Rt29fBAcHY+zY
sVi9ejXXatm0LFxcBo1CrVr10lZWhoiYyNx69YtnDtD3tTnZ8//79/MXPmTBgXbKXL2ZGrn1dH69
ppi7dq1GDp0KJo1a4YBAwagq1apVUryrnpgRlUrynpyc2b94MhUIhlPbL1dH69
atsX//fnTo0AFKbJYc83OzsY3V3xToN3Cjw9Lj9x3o3D999/j27/+++gu
HDh2Cqakpfv/992KdxdW1d69ry8UuXLg/9d6fpH21VdfYcCAAahUqRRJ6
9Ogh/QJ+nUEPHqFmzZZr4+OOP4erqipVpVqqFuDgwDwYM4c+YMFi9j9euDWdnZxw8eLDEHyA3Y
MAATJ8+Hb169KKECROkJf/169+/+cRhRkTdZPJvHnWFQDdlMHWfx/ZmYeDBBKAoGjJLBx7jauE
JoamqKdu3aYeHChZOgvZZ+H772HAwcOIckgQpHHh/zxwk2czPyAWXiAUd9oB6SUQkPDdZgwoMzPxn//8B1Z
WVrLif9OmTVVizZg169eqFOnXq4NGjR/4DU1NTdO/eHQDQvn17jBkzBoGYiNjUWXl1Q
qVIlXL58GTt27MDy5cx8ccfS/msXbsW33zzDerWrQsrKyvypfbU7d+4gLi5OWqJGxi9I/3q
mW2ubm5ok6dOqJOnToiJydHCCHE4sVKmSeO+998QHH3wgfvnllwLHX758XDfp+78S7U
o6fOzYsSLP/fIy3EOHDm1WiMzMyEoaGhqFOnjhfLiIioqSxSUkJAgAolGjRrL2bQ775RgA
QX3/9dbGugY+PPT5FLnuvUqSPFxcTEiN69e/38pHj1qxMw9RsR0Z0JuJ5yoY7HtHZQB4vHJ3VXVlbE
6+PHj4Wfn5wsLAQVapUET8YUnT17/92hXQ92C2PLfePjKTUw1Kj+eKyJgTKwH5rGuu/Kepx

OHv7y9cXV2FiYmJqFy5snB1dRVr1qwp1vVZsmSJqFKlSoGl0UWd+0UHDhwQzs7OQqlUigYNGo
iffvqpTJaJ529KpVLY2NiIzp07i+XLlwuNRlPgmFu3bolevXoJc3NzYWZmJvr27Stu375dYHm
+EEVf9927d4smTZoIQ0NDUatWLbFgwQKxYcMGWczZs2fFwIEDhb29vVCpVMLKykp88MEHBX6O
hRBi/fr1wt3dXRgZGQkTExPh4uIiPv/8c3H79m0pJjU1VXh7ewsTExMBQLZkfO3atcLY2LjQ+
dI/k0KIMnjjj4hIS7GxsXBzc8NPP/0kPVYpbxkZGahduzYWLlworYqjisXNzQ0dOnTA0qVLyz
sVqiD4Dg4RlZunT58WaFu2bBn09PRK5eXo0mJmZobPP/8cixYtKtYqI3q7QkJCcPnyZQQEBJR
3KlSB8A4OEZWbOXPmIDo6Gh07doSBgQH27duHffv2YfTo0fj3v/9d3ukR0TuMBQ4RlZvQ0FDM
mTMH58+fR2ZmJuzt7TF06FB89dVXr/y8ICKi12GBQ0RERDqH7+AQERGRzmGBQ0RERDqnXB9yB
wYGYufOnbh48SKMjIzQunVrLFiwAA0aNJinj17hqlTp2Lr1q3IysqCWq3GmjVrYG1tLcUkJy
fj008/xaFDh1ClShX4+PggMDBQ9gz/8OHDmDJlChISEmBnZ4cZM2Zg+PDhsnxWr16NRYsWITU
1Fa6urli5ciVatGhRrLnk5eXh9u3bMDEx4Re9ERERlYAQAo8ePYKtre0bfznwi4OWG7VaLTZu
3Cji4+NFbGys6N69u7C3txeZmZlSzNixY4WdnZ0ICwsTUVFRolWrVqJ169ZSf05OjnB2dhZeX
l4iJiZG7N27V1haWoqAgAAp5tq1a8LY2FhMmTJFnD9/Xqxcu VLo6+uLkJAQKWbr1q1CqVSKDR
s2iISEBDFq1Chhbm4u0tLSijWXmzdvFvlBady4cePGjRu31283b94sheribxXqJeO7d+/Cyso
KR44cQbt27ZCRkYHq1atjy5Yt0kd1X7x4EY0aNUJkZCRatWqFffv24YMPPsDt27eluzrr1q3D
9OnTcffuXSiVSkyfPh1//PEH4uPjpjpXMNGDAADx8+REhICACgZcuW+Ne//iV9DH5eXh7s7Ozw2
Wef4Ysvvnht7hkZGA3N8fNmzdhampa2peGiIhIZ2k0GtjtZ2eHhw4cwMzMrlTEr1DrM/C/Ds7
CwAPD3F6dlZ2fLvr24YcOGsLe3lwqcyMhIuLi4yB5ZqdVqfPrpp0hISICbmxsiIyYMLfAOyWq3
GpEmTAADPnz9HdHS07EOi9PT04OXlhcjIyEJzzcrkKn1J36NHjwD8/T0vLHCIiIhKrjRf8agw
Lxnn5eVh0qRJaNOmDZydnzQEAqampUCqVMDc3l8VaW1sjNTVVinmxmxTvz97VXFo0Hp0+Rn
p6O3NzcQmPyx3hZYGAgzMzMpM3Ozk67iRMREVGpqzAFjp+fH+Lj47i5OWlgYbGxsp5vT
p07Lx0tLSpL78/81vezHG1NQURkZG0NfXh76+fqEx+WO8TKVSaVSaTdhIiIiKlPlegdHCIHx
48fjt99+Q3h4OBwdHWX97u/uuqFSpEsLCwqS2xMRE3XGZ4eqitWrVxfan5KSIts2bNGv9rx0R5WZGv3iGrJB
WpqisaNG0sxL46RH5M/hlKphLu7O2zYYWPv3OU2FmZiYOHz4sUl
JSpO3JkydSzNixY4W9vb0IDw8XUVFRwsPDh3h4eEj9+cvEoEu3Tp1ImJjY0VISIioXr16ocvE/f3
9xYULF8Tq1asLXSauUqlEUFCQOH/+vBg9erQwNzeXraZ4lYyMDAHQ4mALpQEPRPI1Vbb1arIxYjHiJXlZ+Nb1ka1DcTfVYsLiNH3a3RHtKr
79ByLXBQxDr4jRs3SjFPnz4V48aNE1WrVhXGxsaiV69eIiUlRTb09aAlcwPUVmfEYJr9z9z8Ch
6dOFdnZ2bKYQ4cOiaZNmwqlUilq164e0+lStXCnt7e6FUKkWLFi3EyZMniz0XFjhERETaKY
vfoRXqc3DeZRqNBmZmZsjIyOAycSIohIoi9+hFWYVFREREVFpYYFDEREOocFDhEREemcvE
5OFT6kpOTkZ6eXuLjLC0tYW9vXYZEREREvT0scHRQcnIyGjZqgnZqhKdPnpT4WCgsWYEVFYx9HcHH9H2HQg2OCV2FIWYNlUMmFSOcQQS1aqGMmb0LCY7ZpmHd1
vdNY4Oig9PR0PH3yBOO+Ww/bOvWLfdztq5ewZtpopKens8AhIqJ3GgscHWZbpz4cnZqWdxpER
ERvHV8yJiIiIp3DAoeIiIh0DgscIiIi0jkscIiIiEjnsMAhIiIincMCh4iIiHQOCxwiIiLSOS
xwiIiISOewwCEiIiKdwwKHiIiIdA4LHCIiItI5LHCIiIhI57DAISIiIp3DAoeIiIh0DgscIiI
i0jkscIiIiEjnsMAhIiIincMCh4iIiHQOCxwiIiLSOSxwiIiISOeUa4ETERGBHj16wNbWFgqF
Art27ZL1KxSKQrdFixZJMbVq1SrQHxQUJBsnLi4Onp6eMDQ0hJ2dHRYuXFgglx07dqBhw4YwN
DSEi4sL9u7dWyZzJiIiorJXrgXO48eP4erqitWrVxfan5KSIts2bNgAhUKBPn36yOLmzp0ri/
vss8+kPo1Ggy5dusDBwQHR0dFYtGgRZs+ejR+++EmLmzm/ro0UHHb4NqYqNrhaNWpq36n1t3zOzA7yLmzp0ri/
vss8+kPo1Ggy5dusDBwQHR0dFYtGgRZs+ejfXr10sxJ06cwMCBA+Hr64uYmBj07NkTPXv2RHx
8fNlMnIiIiMqUQXmevFu3bujWrVuR/TY2NrL93bt3o0OHDqi6ulpWbmJiUiM2X+rVq5GTk4MGD
BgwAhL3DaGgoVq1ahXXr1r3JFImImItx
NmzYAKVSCScnJ8TGxmLJkiUYPXP0oAGD58uXo2rUr/P39AQD33a6Q2Qeg/yX7vzY6UU5NZSImcHv2c3NNYnP/NzKSv/9uDuPkWTi/23xL6g/77z9/5b7G/Cy7tIj/KeY+jd+9vdhsHNVFbUlsp
NmzYAKVSCScnJ8TGxmLJkiUYPXo0AGD58uXo2rUr/P39AQD5s1DaGgoVq1ahXXr1r3JFImIi
KgcvDPv4KSlpeGPP/6Ar69vgb6goCBUq1YNbm5uWLRoEXJycqS+yMhItG3bFkqlUmpTq9VITEzEgx
zEgwcPpBgvLy/Zeajs2Fg1GpGRkUXmW25WWNXIvanQQYIiFLpIT/1RSLWBBASTGJtRJw8gSXDkI7mb3hKyZCZjQuiyvemwTgBr1p8LZh6+5
Zs2awsLDAiRMnEBAQgJSUFCxZsgQAkJqaWkZGRkSEzoobbqa8IKB8qQaZ8Ks6w6HROHOtGSyaGaWmQ+fKpGUR1
mDNnTmlMjYiIiErZO1PgbNiwAYMHD4ahoaZsfcqUKdKfHi+wMlB9AagzAUsATrRCFf4m4k5IGC
JCdW6PRwM7OrszOR0RERMX3ThQ4R48eReJiIrZt2/ba2JYtWyInJwfXr1+HjlzgnIFadI5RAwLeGYAZ20c8VSOOIS9WLZ/wOrvrT10K+KnL3fFnmxhA1yMQi/XGdBqRpfd+0wkPuY9Bgn8EqpSeRWR8
lM/n7+eztFxRT1Xg8AqFSqMi2giIiISHvvxDs4//3vf+Hu7g5XV9fXsbxKJPTw9WVlYAAA8
PD0RERCA7O1uKCQ0NRYMGDVC1alUpJiwsDZOaGgoPDw8SnEWRERE9LaUa4GTmZmJJ2NhYxMbG
AgCSkpIQGxuL5ORkUajOWDHjh345JNPChwfGRmJZcuW4c8//8S1a9ewefNmTJ48GUEDB8o
ZSpMAIMMDAbJXjGo7vKiOW6F/5ZrOQh/O2CMf/2/8zdXvfIfrXhOxrnTiv+mWU2rzrVcmMwl/
mjQICiVSvj6+iIhIQHbtm3D8uXLZY+2PvvsM4SEhGDx4sW4ePEiZs+ejaioKIwfP/5NLoOKhsz9p73o6iDtJRmbrW9YhRgtKW7Yw/pdW9nb8Pyrb6mFqedSmcuvuBGgUTE/W
IyUa6PqKKiotCxY0dpP7/w8PHxQ3BwMABg69atEEJg4MCBBR6vUqmwdetWzJ49G1lZWXB0dMTk
yZNlxYu5uTkOHDgAPz8/uLu7w9LSEjNnzpSWiANA69atsWXLFsyYMQNffvkl6tWrh127dpXJZ+GQ
/7zbtGmTJkwSP0+v4XaaSbjpGvLYgN+Rs7Glzk3xmqmEn8ybgRnPnzAxm8HESvzZDuyRVtlHfm9Fe
vpNmQ8DpqGPT2CE+ZUp7SqkWvfMLpgzIjNGsSOOwbq3QgeTlGRRkx7KrmY0y+GcJeTmDefbIFPBb76k4UiQ+Ysk0wD5Uu7pWwjHiLX6dbbwCgEBF6M8DyZ7uNjQXmQLQOXddHT7pOc49ykk3zpjwVGbh77w0pJIWy
GSEAl1E+N9EvOVa0nKW+mkV9cA6lRX1qJuwn6MirAtttsy+mSqKrt8QBxChOCPXyBo7f4ixrk

urli9enWRMV27dkVKSoq0/fzzz7L+wYMHIyEhAaGhodizZw8iIiIwevRoqV+j0aBLly5wcHBA
dHQ0Fi1ahNmzZ2P9+vVSzIkTJzBw4ED4+voiJiYGPXv2RM+ePREfH1/6kyYiIqIyZ1CeJ+/Wr
Ru6dev2yhiVSgUbG5tC+y5cuICQkBCcOXMGzZs3BwCsXLkS3bt3x3fffQdbW1ts3rwZz58/x4
YNG6BUKuHk5ITY2FgsWbJEKoSWL1+Orl27wt/fHwAwb948hIaGYtWqVVi3bl0pzpiIiIjehgr
/Ds7hw4dhZWWFBg0a4NNPP8W9e/ekvsjISJibm0vFDQB4eXlBT08Pp06dkmLatwWsHpVIpxajV
aiQmJuLBgwdSjJeXl+y8arUakZGRReaVlZWFjUYj24iIiKhiqNAFTteuXfHDDz8gLCwMCxYsw
JEjR9CtWzfk5uYCAFJTU2FlZSU7K2tZTH5+K2tZTH5+6+Lye8vTGBgIMzMzKTNzs
7uzSZLREREpaZcH1G9zoABA6Q/u7i4oEmTJqhTpw4OHz6MTp06lWNmQEBAAKZMmSLtazQaFjl
EREQVRIW+g/Oy2rVrw9LSEleuXAEA2NjY4M6dO7KYnJwc3L9/X3pvx8bGBmlpabKY/P3XxRT1
7g/w97tBpqamso2IiIgqhneqwLl16xbu3buHGjVqAAA8PDzw8OFDREdHSzHh4eHIy8tDy5Ytp
ZiIiAhkZ2dLMaGhoWjQoAGqVq0qxYSFhcnOFRoaCg8Pj7KeEhEREZWBci1wMjMzERsbi9jYWA
BAUlISYmNjkZycjMzMTPj7++PkyZO4fv06wsLC8NFHH6Fu3bpQq9UAgEaNGqFr164YNWoUTp8
+jePHj2P8+PEYMGAAbG1tAQCDBg2CUqmEr68vEhISsG3bNixfvlz2eGnixIkICQnB4sWLcfHi
RcyePRtRUVEYP378W78mRERE9ObKtcCJioqCm5sb3NzcAABTpkyBm5sbZs6cCX19fcTFxeHDD
z9E/fr14evrC3d3dxw9ehQqlUoaY/PmzWjYsCE6deqeqE7t27o23btrLPuDEzM8OBAweQlJQEd3
d3TJ06FTNnzpR9Vk7r1q2xZcsWrF+/Hq6urvjll1+wa9cuODs7v72LQURERKWmXF8y7tChA4Q
QRfbv37//tWNYWFhgy5y5Ytr4xp0qQJjh49+sqYvn37om/fvq89HxEREVV879Q7OERERETFwQKH
iIiIdA4LHCIiItI5LHCIiIhI57DAISIiIp3DAoeIiIh0DgscIiIi0jkscIiIiEjnsMAhIiIin
cMCh4iIiHQOCxwiIiLSOSxwiIiISOewwCEiIiKdwwKHiIiIdA4LHCIiItI5LHCIiIhI57DAIS
IiIp3DAoeIiIh0DgscIiIi0jkscIiIiEjnsMAhIiIincMCh4iIiHQOCxwiIiLSOSxwiIiISOe
wwCEiIiKdwwKHiIiIdA4LHCIiItI55VrgREREoEePrC1tYVCocCuXbukvuzsbEyfPh0uLi6o
XLkybbG1tMWzYMNy+fVs2Rq1ataBQKGRnp4wNDSEnZ0dFi5cWCCXHTt2oGHDh
jA0NISLiwv27t1bJnMmIiKisleuBc7jx4/h6uqK1atXF+h//hpnz57Fzp07kZ
iYiA8//LBA7Ny5c5GSkiJtn332mdSXdn39Sn0WjQpUsXODg4IDo6GosWLcLs2bOxfv16KebEiRMYOHA
gfH19ERMTg549e6Jnz56Ij48vm4kTERFRmTIoz5N369YN3bp1N3bp1K7TPzMwMoaGhsrFVq1ahRYsWSW
SE5Ohr29vdRuYmICGxubQsfZvHkznj9/jg0bNkCpVMLJyQmxsbFYsmQJRo8eDQBYvnw5unbtC
n9/fwDAvHnzEBoailWrVmHdunHzEBoailWrVmHdunWlMVUiIJ6i96pd3AyMjKgUChgbm4aw8KCkK1atXg5uaGRY
sWIScnR+qLJjIxEu3btboFQqpTa1Wo3ExEQ8ePBAivHy8pKNq1arERkZWWQeWVlZ0Gg0so2IiIgqhneqwLl16xbu3buHGjVqAAA8PDzw8OFDREdHSzHh4eHIy8tDy5Ytp
ZiIiAhkZ2dLMaGhoWjQoAGqVq0qxYSFhcnOFRoaCg8Pj7KeEhEREZWBci1wMjMzERsbi9jYWA
BAUlISYmNjkZycjMzMTPj7++PkyZO4fv06wsLC8NFHH6Fu3bpQq9UAgEaNGqFr164YNWoUTp8
+jePHj2P8+PEYMGAAbG1tAQCDBg2CUqmEr68vEhISsG3bNixfvlz2eGnixIkICQnB4sWLcfHi
RcyePRtRUVEYP378W78mRERE9ObKtcCJioqCm5sb3NzcAABTpkyBm5sbZs6cCX19fcTFxeHDD
z9E/fr14evrC3d3dxw9ehQqlUoaY/PmzWjYsCE6deqeqE7t27o23btrLPuDEzM8OBAweQlJQEd3

YEePXrA1tYWCoUCu3btkvULITBz5kzUqFEDRkZG8PLywuXLl2Ux9+/fx+DBg2Fqagpzc3P4+v
oWWMkVFxcHT09PGBoaws7ODgsXLiyQy44dO9CwYUMYGhrCxcUFe/fuLf4FISIiogpFqwKnZs2
aOHfuHL788ktMnjwZbm5uCAoKQkxMDKysrIo9zuPHj+Hq6orVq1cX2r9w4UKsWLEC69atw6lT
p1C5cmWo1Wo8e/ZMihk8eDASEhIQGhqKPXv2ICIiAqNHj5b6NRoNunTpAgcHB0RHR2PRokWYP
Xs21q9fL8WcOHECAwcOhK+vL2JiYtCzZ0/07NkT8fHxWlwdIiIiKm9avWQMAAYGBlp/pH++bt
26oVu3boX2CSGwbNkyzJgxAx999BEA4IcffoC1tTV27dqFAQMG4MKFCwgJCcGZM2fQvHlzAMD
KlSvRvXt3fPfdd7C1tcXmzZvx/PlzbNiwAUqlEk5OToiNjcWSJUukQmj58uXo2rUr/P39AQDz
5s1DaGgoVq1aVeSS96ysLGRlZUn7Go3mja4FERERlR6tCpwffvjhlf3Dhg3TKpkXJSUlITU1F
V5eXlKbmZkZWrZsicjISAwYMACRkZEwNzeXihsA8PLygp6eHk6dOoVevXohMjIS7dq1k634Uq
vVWLBgAR48eICqVasiMjISU6ZMkZ1frVYXeGT2osDAQMyZM+eN50lERESlT6sCZ+LEibL97Ox
sPHnyBEqlEsbGxqVS4KSmpgIArK2tZe3W1tZSX2pqaoFHYgYGBrCwsJDFODo6Fhgjv69q1apI
TU195XkKExAQICuKNBoN7OzsSjJFIiIiKiNaFTgPHjwo0Hb58mV8+umn0mMeXadSqaBSqco7D
SIiIiqE1t9F9bJ69eohKCiowN0dbdnY2AAA0tLSZO1paWlSn42NDe7cuSPrz8nJwf3792UxhY
3x4jmKisnvJyIiondLqRU4wN+Ph27fvl0qYzk6OsLGxkb21Q8ajQanTp2Ch4cHAMDDwwMPHz5
EdHS0FBMeHo68vDy0bNlSiomIiEB2drYUExoaigYNGqBq1apSzMtfMREaGiqdh4iIiN4tWj2i
2r17t2xfCIGUlBSsWrUKbdq0KfY4mZmZuHLlirSflJSE2NhYWFhYwN7eHpMmTcI333yDevXqw
dHREV9//TVsbW3Rs2dPaoFHYgYGBrCwsJDFODo6Fhgjv69q1apIgqAK0KnLy8vFI5eYcOHSCEKLJfoV
Bg7ty5mDt3bpExFhYW2LJlyyvP06RJExw9evSVMX379kXfvn1fnTARERG9E0r1HRwiIiKiikC
rOzgvfyjeqyxZskSbUxARERFpTasCJyYmBjExMcjOzkaDBg0AAJcuXYK+vj7NWsmxSkUitLJ
koiIiKgEtCpwevToARMTE2zatElaav3gwQOMGDECnp6emDp1aqkmSURERFQSWr2Ds3jxYgQGB
krFDQBUrVoV33zzTYlWURERERGVBa0KHI1Gg7t37xZov3v3Lh49evTGSREREERG9Ca0KnF69em
HEiBHYuXMnbt26hVu3buHXX3+Fr68vevXXdo5EhEREZWIVu/grFu3DtOmTcOgQYOkvcqlmhwRERGRNt7og/5S
UlKQkpKCevXqoXoXLlyq/8VGIiIiKit0WrAufevXvo1KkT6tevj5y5cqlmhwRERGRNt7og/5S
UlKQkpKCevXqoXoXLlyq/8VGIiIiKit0WrAufevXvo1KkT6tevj7+duyMlJQUA4OvryyXiREREV
O60KnAmT56MSpUqITk5GcbGxlJ7//79ERISUmrJEREREWlDq3dwDhw4gP8796NmzZqy9nr16u
HGjRulkhgRERGRtrS6g/P48WPZnZt89+/fh0qleuOkiIiIiN6EVgWOp6cnfvjhUPPZs2cqMC5du0aatbWlDq
PCxcuRMeOHUstOSIiIiJtaPWIauHChejUqRPqVSMD9+/fR0REBCUlJdSREREWlDq3dwDhw4gP8
qEoveyIiIiItFCiAqdevXpISUnBoUOHAz91QwrVqyAtbV1mSRHREREpI0SvYPz8reF79u3
D48fPy7VhIiIiqelFYvGed7ueAhIiIiqghKVOAoFIoC79jwnRsiIiKqaEr0Do4QAsOHD5e+U
PPZs2cYO3ZsgVVUO3fuLL0MiYiIiEqoRHdwfHx8YGVlBTMzM5mZmWHIkCGwsbGBvr4+zVrJ
3bh4cOHEydOnDp1aqkmSURERFQSWr2Ds3jxYgQGBkxY8ewdOlSUYt26dXB0dMTNmzdRVlZG
i3pztGLm5+fHwCgQ4cOBfrGjh0rGyM5ORe3t4wNjaGlZUV/P39kZOTI4s5fPgwmjVrBpVKhbVi
3pztGLm5+fHwCgQ4cOBfrGjh0rGyM5ORe3t4wNjaGlZUV/P39kZOTI4s5fPgwmjVrBpVKhbhbp
16yI4OLhU50FERERvT4nu4GzcuLGs8ijSmTNnkJubK+3Hx8ejc+fO3r9Q2atQozJ07V9o3
NjaW/pybmwtvb2/Y2NjgxIkTSElJwbHDVkyBD897//Le1ciIiIiIiEqFVi
8Z5+TkYMOGDTh48OCIiIiIm2UqMC5du0aatbWlDqPCxcuRMeOHUstOSIiIiJtaPWIauHChejUqRPqVSMD9+/fR0REBCUlJdSREREWlDq3dwDhw4gP8qEoveyIiIiItFCiAqdevXpISUnBoUOHAz91QwrVqyAtbV1mSRHREREpI0SvYPz8reF79u3
D48fPy7VhIiIiqelFYvGed7ueAhIiIiqghKVOAoFIoC79jwnRsiIiKqaEr0Do4QAsOHD5e+U
PPZs2cYO3ZsgVVUO3fuLL0MiYiIiEqoRHdwfHx8YGVlBTMzM5mZmWHIkCGwsbGBvr4+zVrJ
3bh4cOHEydOnDp1aqkmSURERFQSWr2Ds3jxYgQGBkxY8ewdOlSUYt26dXB0dMTNmzdRVlZG
i3pztGLm5+fHwCgQ4cOBfrGjh0rGyM5ORe3t4wNjaGlZUV/P39kZOTI4s5fPgwmjVrBpVKhbVi
3pztGLm5+fHwCgQ4cOBfrGjh0rGyM5ORe3t4wNjaGlZUV/P39kZOTI4s5fPgwmjVrBpVKhbhbp
16yI4OLhU50FERERvT4nu4GzcuLGs8ijSmTNnkJubK+3Hx8ejc+fO3r9Q2atQozJ07V9o3
NjaW/pybmwtvb2/Y2NjgxIkTSElJwbHDVkyBD897//Le1ciIiIiIiEqFVi

```
smTYa6enpLHCIiKjUsMChUlXDsS4cnZqWdxpERPQPV6FfMp49ezYUCoVsa9iwodT/7Nkz+Pn5
oVq1aqhSpQr69OmDtLQ02RjJycnw9vaGsbExrKys4O/vj5ycHFnM4cOH0axZM6hUKtStWxfBw
cFvY3pERERURip0gQMATk5OSElJkbZjx45JfZMnT8bvv/+OHTt24MiRI7h9+zZ69+4t9efm5s
Lb2xvPnz/HiRMnsGnTJgQHB2PmzJlSTFJSEry9vdGxY0fExsZi0qRJ+OSTT7B///63Ok8iIiI
qPRX+EZWBgQFsbGwKtGdkZOC///0vtmzZgvfffx8AsHHjRjRq1AgnT55Eq1atcODAAZw/fx4H
Dx6EtbU1mjZtinnz5mH69OmYPXs2lEol1q1bB0dHRyxevBgA0KhRIxw7dgxLly6FWq1+q3MlI
iKi0lHh7+BcvnwZtra2qF27NgYPHozk5GQAQHR0NLKzs+Hl5SXFNmzYEPb29oiMjAQAREZGws
XFBdbW1lKMWq2GRqNBQkKCFPPiGPkx+WMUJSsrCxqNRrYRERFRxVChC5yWLVsiODgYISEhWLt
2LZKSkuDp6YlHjx4hNTUVSqUS5ubmsmOsra2RmpoKAEhNTZUVN/n9+X2vitFoNHj69GmRuQQUG
BsLMzEza7Ozs3nS6REREVEoq9COqbt26SX9u0qQJWrZsCQcHB2zfvh1GRkbkQEBCAKVOmS
PsajYZFDhERUQVRoQucl5mbm6N+/fq4cuKOnfujOfPn+Phw4eyuzhpaWnSOzs2NjY4ffbI
z8VVYvxry88iotLQ2mpqavLKJUKhVUKlVpTOuVkpOTkZ6eXqJjtPk0YSIiIl3yThU4mZmZuHr
1KoYOHQp3d3dUqlQJYWFh6NOnDwAgMTERycnJ8PDwAAB4eHhg/vz5uHPnDqysrAAAoaGhMDU1
RePGjaWYvXv3ys4TGhokSr459nPS/ljIiIiN4NFbrAmTZtGnr06AEHB
wfcvn0bs2bNgr6+PgYOHAgzMzP4+vpiypQbQpsLCwgKmpKT777DN4eHigVatWAIAuXbqqcePGGD
p0KBYuXIjU1FTMmDEDfn5+0t2XsWPHYtWqVfj8888xcuRIhIeHY/v374fT0dD/g4X4/v27fjjjjz/Kc+oAgPT0dDx
98gTjvlsP2zr1i33cuSOh2LFsfoHP+yEiIvqnqNAFzq1atzBw4EDcv3/fhWHWwHXAhlyBHU1Zr+KN8KY+I2K1cJaO2k1ATHGdYc+fGrKaVH16X28jvSWaF5XmSyzpYP6Tw9d8lbgHsHY1oSlR
zqgmfRQBBCNQjRbYfWQmmUFQQrnYBSMXzRGp3DNmnMp1aSsoQx52cgLL4Uo0aDpUPGRmgaSMaBBX
Z2xlRCUbHAzrwMQvrE9Imx+AfXkzqoL7E12XnMadXe5BPVVKewDqxNETTDQHGzgRp9
HWqcF2Z72bUHLOSIBRvqGZZWchDCZnQGqnVhqgj3N2FwaVNvmFGpD/WQBCkq+kOffJUl2CX1XcTPDB
pZIfahpDXoVhSIXQrZnnJwOPgsqThv6FGPQ/Ff3vJ2mYkyOWTkMGGB7Nt1+KaKQ3+mVYNPGV
```

73DlEol3N3dERYWJrXl5eUhLCwMHh4e5ZgZERERlRTv4LxgypQp8PHxQfPmzdGiRQssW7YMjx
8/llZVERER0buBBc4L+vfvj7t372LmzJlITU1F06ZNERISUuKVGqVBpVJh1qxZWn1C7LuM8+a
8/wn+ifP+J84Z4LzLc94KIUpzTRYRERFR+eM7OERERKRzWOAQERGRzmGBQ0RERDqHBQ4RERHp
HBY4FdTq1atRq1YtGBoaomXLljh9+nR5p1RsgYGB+Ne//gUTExNYWVmhZ8+eSExMlMU8e/YMf
n5+qFatGqpUqYI+ffoU+JDF5ORkeHt7w9jYGFZWVvD390dOTo4s5vDhw2jWrBlUKhXq1q2L4O
Dgsp5esQQFBUGhUGDSpElSm67O+a+//sKQIUNQrVo1GBkZwcXFBVFRUVK/EAIzZ85EjRo1YGR
kBC8vL1y+fFk2xv379zF48GCYmprC3Nwcvr6+0nex5YuLi4OnpycMDQ1hZ2eHhQsXvpX5FSY3
Nxdff/01HB0dYWRkhDp16mDevHmy79HRhXlHRESgR48esLW1hUKhwK5du2T9b3OOO3bsQMOGD
WFoaAgXFxfs3bu31Oeb71Xzzs7OxvTp0+Hi4oLKlSvD1tYWw4YNw+3bt2Vj6Nq8XzZ27FgoFA
osW7ZM1l6h5l1qX/pApWbr1q1CqVSKDRs2iISEBDFq1Chhbm4u0tLSyju1YlGr1WLjxo0iPj5
exMbGiu7duwt7e3uRmZkpxYwdO1bY2dmJsLAwERUVJVq1aiVat24t9efk5AhnZ2fh5eUlYmJi
xN69e4WlpaUICAiQYq5duyaMjY3FlClTxPnz58XKlSuFvr6+CAkJeavzfdnp06dFrVq1RJMmT
cTEiROldl2c8/3794WDg4MYPny4OHXqlLh27ZrYv3+/uHLlihQTFBQkzMzMxK5du8S5c+fEhx
9+KBwdHcXTp0+lmK5duwpXV1dx8uRJcfToUVG3bl0xcOBAqT8jI0NYW1uLwYMHi/j4ePHzzz8
LIyMj8e9///utzjff/PnzRbVq1cSePXtEUlKS2LFjh6hSpYpYvny5FKML8967d6/46quvxM6d
OwUA8dtvv8n639Ycjx8/LPvT19cXChQvF+fPnxYwZM0SlSpXEn3/++dbn/fDhQ+Hl5SW2bdsmL
l68KCIjI0WLFi2Eu7u7bAxdm/eLdu7cKVxdXYWtra1YunSprK8izZsFTgXUokUL4efnJ+3n5u
YKW1tbERgYWI5Zae/OnTsCgDhy5IgQ4u9/ICpVqiR27NghxVy4cEEAEJGRkUIv/+i6enpidT
UVClm7dq1wtTUVGRlZQkhhPj888+Fk5OT7Fz9+/cXarW6rKdUpEePHol69eqJ0NBQ0b59e6nA
0dU5T58+XbRt27bI/ry8PGFjYyMWLVoktT18+FCoVCrx888/CyGEOH/+vAAgzpw5I8Xs27dPK
BQK8ddffwkhhFizZo2oWrWqdB3yz92gQYPSnlKxeHt7i5EjR8raevfuLQYPHiyE0M15v/wL72
3OsV+/fsLb21uWT8uWLcWYMWNKdY6eEELo971u3bon33ntPxMfHCwc
HB1mBU9HmzUdUFczz588RHR0NLy8vqU1PTw9eXl6IjIwsx8y8yO15GRAQCwsLAAAERHRyM7O1s2
x4YNG8Le3l6aY2RkJFxcXGQfsqhWq6HRaJCQkCDFvDhGfkx5Xic/Pz94e3sXyEtX57x79240b
94cffv2hZWVFdzc3PCf//xH6k9KSkJqaqosZzMzM7Rs2VI2b3NzczRv3lyK8fLygp6eHk6dOi
XFtGvXDkqlUopRq9VITEzEgwcPynqaBbRu3Rrh4eG4dOkSAODcuXM4duwYunXrBkB35/2itzn
HivZz/7KMjAwoFArp+wh1dd55eXkYOnQo/P394eTkVKBfF+bNAqcCunfvHnJzcwt8OaS1tTV
SU1PLKautJTHKSBpu9ds6Qt9qc+D6FqiH76eTIuyMBoEQEU
...

```
EoGPHjvDz89P63MXxJnMrCxUtH6LSwgKH6B3n5eUFGxsbBAYGFhkze/bsAo9rli1bhlq1akn7
+XcEvv32W1hbW8Pc3Bxz585FTk4O/P39YWFhgZo1a2Ljxo0Fxr948SJat24NQ0NDODs748iRI
7L++Ph4dOvWDVWqVIG1tTWGDh2K9PR0qb9Dhw4YP348Jk2aBEtLS6jV6kLnkZeXh7lz56JmzZ
pQqVRo2rQpQkJCpH6FQoHo6GjMnTsXCoUCs2fPLnSccePGQaFQ4PTp0+jTpw/q168PJycnTJk
yBSdPnpTFpqeno1evXjA2Nka9evWwe/duqS83Nxe+vr5wdHSEkZERGjRogOXLl8uOz7+u8+fP
h62tLRo0aAAA+PHHH9G8eXOYmJjJAxsYGgwYNwp07d2THJiQk4IMPPoCpqSlMTEzg6emJq1evY
vbs2di0aRP+97//QaFQQKFQ4PDhwwCAmzdvol+/fjA3N4eFhQU++ugjXL9+/bX5rFmzBvXq1Y
OhoSGsra3x8ccfF3rtiN4VLHCI3nH6+vr49ttvsXLlSty6deuNxgoPD8ft27cRERGBJUuWYNa
sWfjggw9QtWpVnDp1CmPHjsWYMWMKnMff3x9Tp05FTEwMPDw80KNHD9y7dw8A8PDhQ7z//vtw
c3NDVFQUQkJCkJaWhn79+snG2LRpE5RKJY4fP45169YVmt/y5cuxePFifPfdd4iLi4NarcaHH
36Iy5cvAwBSUlLg5OSEqVOnIiUlBdOmTSswxv379xESEgI/Pz9Urly5QL+5ublsf86cOejXrx
/i4uLQvXt3DB48GPfv3wfwd8FVs2ZN7NixA+fPn8fMmTPx5ZdfYvv27bIxwsLCkJiYiNDQUOz
ZswcAkJ2djXnz5uHcuXPYtWsXrl+/juHDh0vH/PXXX2jXrh1UKhXCw8MRHR2NkSNHCnB9Om
TUO/fv3QtWtXpKSkICUlBa1bt0Z2djbUajVMTExw9OhRHD9+HFWqVEHXrl1ldd2pezicqKgoTJ
kzA3LlzkZiYiJCQELRr167Q/w+I3hkl/v5xIqowfHx8xEcffSSEEKJVq1Zi5MiRQgghfvvtN/
HiX+9Zs2YJV1dX2bFLly4VDg4OsrEcHBxEbm6u1NagQQPh6ekp7efk5IjKlSuXLn3/+WQghRFJ
SkgAggoKCpJjs7GxRs2ZNsWDBAiGEEPPmzRNdunRndunSRnfvmzZsCgEhMTBRCCNG+fXvh5ub22vna
2tqK+fPny9r+9a9/iXHjxx4n7rq6uYtasWUWOcerUKQFA7Ny587XnAyBmzJgh7WdmZgoAYt++f
UUe4+fnJ/r06SPt+/j4CGtra5GVlfXKc505c0YAEI8ePRJCQCBEQECACAcHR3F8+fPC41/8f/7fD
/++KNo0KCByMvLk9qysrKEkZGR2L9/f5H5/Prrr8LU1FRoNJpX5kj0LuEdHCIdsWDBAmzatAk
XLlzegwnJyfo6f3/fxasra3h4uIi7evr66NatWoFHqV4eHhIfzYwMEDz5s2lPM6dO4dDhw6h
SpUq0tawYUMAf78vk8/d3f2VuWk0Gty+fRtt2rSRtbdp06ZEcxZCFDsWAJo0aSL9uXLlyjA1N
ZXNf/Xq1XB3d0f16tVRPUoVrF+/HsnJybIxXFxcoFQqZW3R0dHo0aMH0aMH7O3tYWJigvbt2wd
xsbCw8PT1RqVKlYud67tw5XLlyBSYmJtK1trCwwLNnz2TX+uV8OnfuDAcHB9SuXRtDhw7F5s2
b8eTJk4cPHyInJ0esWAJo0aOFqIAoFQqkZubK2tr1qwZLl
++ODCsrqwLX2szM7JU5GRgYwMvLCwsXLkRcXByX7+O8PDw186FqKJigUOkQ1xcXDB48GCsWLF
C1t6hQwfcvXsXxCxcuxNWrV7F69Wrs9Wrs27ev1M67evVq/Pbbb7777rvvVq5cKSUlJcXGxs
+/fvvyY8SIEQV+Qb+Ov78/FixYgG3btiExMRERQV+Qb+Ov78/FixYgGz5gyuXr2K/XKD/4cOHxR7r+PHjaN26NcN
iVSqxcuRLXrl3D7t27ZWZiYCACoVasW4uLikJiYiPT0dGRkZSQq9VYvXo1Hj9+XKD/4cOHxR7r+PHjaN26NcN
YiMEzg6emJq1evvvvuu+/WrVsnJcXGxs+/fvvyY8SIEQV+Qb+Ov78/FixYgG3btiExMRERQV+Qb+Ov78/FixYgGz5gyuXr2K/XKD/4cOHxR7r+PHjaN26NcN
H5CXlyetsCJ6F7HAIdIxc+fOLfAIqVGjRlizZg1Wr14NV1dXnD59+utAVRtoKCgpCUFAQXF1dc
ezYMezevRuRuWlpYAIN11yc3NRZcuXeDi4oJJkybB3Nxc3Nxc9r5PcUyYMAFTpkzB1KlT4eLiYmQ
zevRv16tUr0Ti1a9fG2bNn0xOzs7o3LkzsCqaBSipG/cERERERVVwvINDRERREOocFDhERERe
kcFjhERESkc1jgEBERkc5hgUNEREQ6hwUOERERKRzWOAQERGRzmQBGBQ0RERD0HBQ4OERERKRzWOERERBQ4
RERHpHBY4REREpEpHP+H/xb9bjlz/+PAAAAAElFTkSuQmCC\n"
```


```
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "\n",
            "positive sample:\n",
            " One of the other reviewers has mentioned that after
watching just 1 Oz episode you'll be hooked. They are right, as this is
exactly what happened with me.<br /><br />The first thing that struck me
about Oz was its brutality and unflinching scenes of violence, which set
```

```
in right from the word GO. Trust me, this is not a show for the faint hearted or timid. This show pulls no punches with regards to drugs, sex or violence. Its is hardcore, in the classic use of the word.<br /><br />It is called OZ as that is the nickname given to the Oswald Maximum Security State Penitentary. It focuses mainly on Emerald City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy dealings and shady agreements are never far away.<br /><br />I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. Forget pretty pictures painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was surreal, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence. Not just violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle class inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is uncomfortable viewing....thats if you can get in touch with your darker side.\n",
            "\n",
            "negative sample:\n",
            " Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time.<br /><br />This movie is slower than a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.<br /><br />OK, first of all when you're going to make a film you must Decide if its a thriller or a drama! As a drama the movie is watchable. Parents are divorcing & arguing like in real life. And then we have Jake with his closet which totally ruins all the film! I expected to see a BOOGEYMAN similar movie, and instead i watched a drama with some meaningless thriller spots.<br /><br />3 out of 10 just for the well playing parents & descent dialogs. As for the shots with Jake: just ignore them.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Preprocessing"
      ],
      "metadata": {
        "id": "B1AgViGtbRob"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import re\n",
        "import nltk\n",
        "nltk.download('stopwords')\n",
        "from nltk.corpus import stopwords\n",
        "\n",
        "stop_words = set(stopwords.words('english'))\n",
        "\n",
```

```
        "def preprocess_text(text):\n",
        "    text = text.lower()                            # Lower case\n",
        "    text = re.sub(r'<.*?>', '', text)              # Remove HTML
tags\n",
        "    text = re.sub(r'[^a-z\\s]', '', text)         # Remove
punctuation & numbers\n",
        "    text = re.sub(r'\\s+', ' ', text).strip()     # Remove extra
whitespace\n",
        "    text = ' '.join(word for word in text.split() if word not in
stop_words)  # Remove stopwords\n",
        "    return text\n",
        "\n",
        "df['clean_review'] = df['review'].apply(preprocess_text)\n",
        "\n",
        "print(df[['review', 'clean_review']].head(3))\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Ey61o-lIcGv5",
        "outputId": "bce1bed4-da1f-4899-8ade-037cbd493212"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "[nltk_data] Downloading package stopwords to
/root/nltk_data...\n",
            "[nltk_data]   Unzipping corpora/stopwords.zip.\n"
          ]
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "                                                review  \\\n",
            "0  One of the other reviewers has mentioned that ...   \n",
            "1  A wonderful little production. <br /><br />The...   \n",
            "2  I thought this was a wonderful way to spend ti...   \n",
            "\n",
            "                                        clean_review  \n",
            "0  one reviewers mentioned watching oz episode yo...  \n",
            "1  wonderful little production filming technique ...  \n",
            "2  thought wonderful way spend time hot summer we...  \n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "\n",
        "# 1. Preprocess train and test reviews\n",
        "train_df['clean_review'] =
train_df['review'].apply(preprocess_text)\n",
```

```
        "test_df['clean_review'] =
test_df['review'].apply(preprocess_text)\n",
        "\n",
        "# 2. Extract cleaned texts as lists\n",
        "train_texts = train_df['clean_review'].tolist()\n",
        "val_texts = test_df['clean_review'].tolist()\n"
      ],
      "metadata": {
        "id": "MpSrN4dQd5dc"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "Tokenization with BERT"
      ],
      "metadata": {
        "id": "BxNBPRRteRIU"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from transformers import BertTokenizer\n",
        "\n",
        "tokenizer = BertTokenizer.from_pretrained('bert-base-
uncased')\n",
        "def tokenize_function(texts):\n",
        "    return tokenizer(texts, padding=\"max_length\",
truncation=True, max_length=256)\n",
        "\n",
        "train_encodings = tokenize_function(train_texts)\n",
        "val_encodings = tokenize_function(val_texts)\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 272,
          "referenced_widgets": [
            "d5c0880b77804ed9a66f3f9e7be60d94",
            "20e8a4d6cb134f409f780b0e2f4e5525",
            "66c9ea5ab9e7484db9d9fa91e151694f",
            "01e8aeb9d16b452c9d26c471318a15eb",
            "91ec85970c8f4e989dda0845ec7b8275",
            "592c579da11746318a850768fe0213b8",
            "38e9610a0691416a84d9de14a12af7f5",
            "a9113df88e7f4500a8d336bd38a5062c",
            "071d09ab23a1430a9d01cc42a491f7e8",
            "21c091b2eb31493e875e6b2680246ffe",
            "20cb5f55100b471ebd4b913ed2d399f4",
            "90640224e6104123add42a24e6631ca3",
            "4abe3945e7714ad6a5138c4156c1f23a",
            "df56bb7bbd424f2cb7afd07565a225aa",
            "dfe19ce8497d40f9b5360b1384d0e3b6",
            "7faad8b61f0549d6ba365fa8e62deec3",
            "36c82465d74e490c9eb4837d55de5ce5",
```

```
            "621722180acf4129afac57b11cabc519",
            "b6f154bb03e14318bb93bf1b7277330c",
            "cb4e20bcecf946afa5b344386aadb08a",
            "ea3b640fda9e40e4a501373154a983c2",
            "9cb0bc0f6acf41bbb3bb78074ffab0d8",
            "e734413b98bc4835bc9b6136be0648cc",
            "16e900b72d244dbd8e4cc628bbd7f6b5",
            "9d3cab148a9245fcb3a0ccd3895d8cbe",
            "dd7c450f03444e339f2aeb40250b2daf",
            "81eb7ba3f5a645518a0bac0f82bab38f",
            "d8147ead64ac40a1b2cc48b29c996f17",
            "69d2b6323afb41c5b792419f7875da85",
            "df5837a5d7d44e68a62dfbb43fb74a63",
            "d222f4dccef04f7c8e025274cedb71d9",
            "215d1a89ae4d4904a052b5ea71f072c0",
            "90362bc6983d4bb3b5303628bc800e88",
            "2068db549da04ce7aeb21174ce5e34a4",
            "5a646fb35b334749895c90ac702ab697",
            "127d44f53ce7403da9d8ead1b5219608",
            "5d3ade9d6b664c6ba144d8928db12797",
            "b3c4db81eea44982bcf7ed55d05b610a",
            "2c7210afa5d0413c9a534c98387669a9",
            "fd676571ce514ac8893c9950dfccd578",
            "a75d5c05c75c4eae8bf41136f041622a",
            "c5cacf1be5f34ac984ffed981ace5b27",
            "d91f7eab1fcf47e48e1e062a9b57785f",
            "bedce2c1af714598b86eb87de56bd14e"
          ]
        },
        "id": "sGBLdsqCdlqr",
        "outputId": "914c78ac-ed0a-4de6-c31d-39c34f88d126"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/usr/local/lib/python3.11/dist-
packages/huggingface_hub/utils/_auth.py:94: UserWarning: \n",
            "The secret `HF_TOKEN` does not exist in your Colab
secrets.\n",
            "To authenticate with the Hugging Face Hub, create a token in
your settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.\n",
            "You will be able to reuse this secret in all of your
notebooks.\n",
            "Please note that authentication is recommended but still
optional to access public models or datasets.\n",
            "  warnings.warn(\n"
          ]
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "tokenizer_config.json:   0%|          | 0.00/48.0
[00:00<?, ?B/s]"
```

```
            ],
            "application/vnd.jupyter.widget-view+json": {
              "version_major": 2,
              "version_minor": 0,
              "model_id": "d5c0880b77804ed9a66f3f9e7be60d94"
            }
          },
          "metadata": {}
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]"
            ],
            "application/vnd.jupyter.widget-view+json": {
              "version_major": 2,
              "version_minor": 0,
              "model_id": "90640224e6104123add42a24e6631ca3"
            }
          },
          "metadata": {}
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "tokenizer.json:   0%|          | 0.00/466k [00:00<?,
?B/s]"
            ],
            "application/vnd.jupyter.widget-view+json": {
              "version_major": 2,
              "version_minor": 0,
              "model_id": "e734413b98bc4835bc9b6136be0648cc"
            }
          },
          "metadata": {}
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]"
            ],
            "application/vnd.jupyter.widget-view+json": {
              "version_major": 2,
              "version_minor": 0,
              "model_id": "2068db549da04ce7aeb21174ce5e34a4"
            }
          },
          "metadata": {}
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Encode Labels"
```

```
  ],
  "metadata": {
    "id": "Z3fJ_Nj6eclr"
  }
},
{
  "cell_type": "code",
  "source": [
    "label_map = {'negative': 0, 'positive': 1}\n",
    "\n",
    "train_labels = train_df['sentiment'].map(label_map).tolist()\n",
    "val_labels = test_df['sentiment'].map(label_map).tolist()\n"
  ],
  "metadata": {
    "id": "tdWXrXUVec3E"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "markdown",
  "source": [
    "Create Hugging Face Dataset objects"
  ],
  "metadata": {
    "id": "3UZYT2rGekIO"
  }
},
{
  "cell_type": "code",
  "source": [
    "from datasets import Dataset\n",
    "\n",
    "train_dataset = Dataset.from_dict({\n",
    "    'input_ids': train_encodings['input_ids'],\n",
    "    'attention_mask': train_encodings['attention_mask'],\n",
    "    'labels': train_labels\n",
    "})\n",
    "\n",
    "val_dataset = Dataset.from_dict({\n",
    "    'input_ids': val_encodings['input_ids'],\n",
    "    'attention_mask': val_encodings['attention_mask'],\n",
    "    'labels': val_labels\n",
    "})\n"
  ],
  "metadata": {
    "id": "O-ee-VQyek0b"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "markdown",
  "source": [
    "Load BERT Model for Fine-Tuning"
  ],
  "metadata": {
    "id": "FPqW8EnDhCQe"
```

```
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from transformers import BertForSequenceClassification\n",
        "\n",
        "# Load BERT base model with a classification head (2 classes:
positive, negative)\n",
        "model = BertForSequenceClassification.from_pretrained('bert-
base-uncased', num_labels=2)\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 105,
          "referenced_widgets": [
            "3dab77f382a84cd1bfe382a3d1cfac88",
            "7f6b2827fede468c884ccbe04ab220e3",
            "63adaf995f994837876b940a4a780206",
            "393e273f27864d7da5d4a0ce0a84bdaf",
            "720e8083a8084c4fa22bf5d999e1f404",
            "5af5d43331214484b0c91d96f18b6409",
            "8f941a9873064e488c5108156b5f7189",
            "08b1f720830449f4905934800edfecab",
            "290ee96b7c414063aa731338e1dd160f",
            "09eb4ab9535a4f68989a8e6c0bac2b6f",
            "6861748777294114948f65493c8edfdb"
          ]
        },
        "id": "K69KI3nJhCnh",
        "outputId": "004f32dd-9086-4ad4-9d3f-b27d7282c547"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "model.safetensors:   0%|          | 0.00/440M [00:00<?,
?B/s]"
            ],
            "application/vnd.jupyter.widget-view+json": {
              "version_major": 2,
              "version_minor": 0,
              "model_id": "3dab77f382a84cd1bfe382a3d1cfac88"
            }
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "Some weights of BertForSequenceClassification were not
initialized from the model checkpoint at bert-base-uncased and are newly
initialized: ['classifier.bias', 'classifier.weight']\n",
```

```
            "You should probably TRAIN this model on a down-stream task
to be able to use it for predictions and inference.\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Training Configuration"
      ],
      "metadata": {
        "id": "oNEv3vGZhgqP"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "from transformers import Trainer, TrainingArguments\n",
        "from sklearn.metrics import accuracy_score, f1_score\n",
        "import numpy as np\n",
        "\n",
        "def compute_metrics(eval_pred):\n",
        "    logits, labels = eval_pred\n",
        "    predictions = np.argmax(logits, axis=-1)\n",
        "    acc = accuracy_score(labels, predictions)\n",
        "    f1 = f1_score(labels, predictions)\n",
        "    return {'accuracy': acc, 'f1': f1}\n",
        "\n",
        "training_args = TrainingArguments(\n",
        "    output_dir='./results',\n",
        "    num_train_epochs=3,\n",
        "    per_device_train_batch_size=16,\n",
        "    per_device_eval_batch_size=32,\n",
        "    eval_strategy='epoch',\n",
        "    save_strategy='epoch',\n",
        "    logging_dir='./logs',\n",
        "    logging_steps=100,\n",
        "    load_best_model_at_end=True,\n",
        "    metric_for_best_model='f1',\n",
        "    greater_is_better=True,\n",
        "    seed=42,\n",
        "    fp16=True,\n",
        "    report_to=[]\n",
        ")\n",
        "\n",
        "trainer = Trainer(\n",
        "    model=model,\n",
        "    args=training_args,\n",
        "    train_dataset=train_dataset,\n",
        "    eval_dataset=val_dataset,\n",
        "    compute_metrics=compute_metrics,\n",
        ")\n",
        "\n",
        "trainer.train()\n"
      ],
      "metadata": {
        "colab": {
```

```
          "base_uri": "https://localhost:8080/",
          "height": 205
        },
        "id": "1GQVBFvliLrm",
        "outputId": "db993751-a999-4bcf-cf68-96ec8f22f460"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ],
            "text/html": [
              "\n",
              "    <div>\n",
              "      \n",
              "      <progress value='7500' max='7500'
style='width:300px; height:20px; vertical-align: middle;'></progress>\n",
              "      [7500/7500 28:46, Epoch 3/3]\n",
              "    </div>\n",
              "    <table border=\"1\" class=\"dataframe\">\n",
              "  <thead>\n",
              " <tr style=\"text-align: left;\">\n",
              "      <th>Epoch</th>\n",
              "      <th>Training Loss</th>\n",
              "      <th>Validation Loss</th>\n",
              "      <th>Accuracy</th>\n",
              "      <th>F1</th>\n",
              "    </tr>\n",
              "  </thead>\n",
              "  <tbody>\n",
              "    <tr>\n",
              "      <td>1</td>\n",
              "      <td>0.286100</td>\n",
              "      <td>0.261122</td>\n",
              "      <td>0.894500</td>\n",
              "      <td>0.900519</td>\n",
              "    </tr>\n",
              "    <tr>\n",
              "      <td>2</td>\n",
              "      <td>0.184000</td>\n",
              "      <td>0.303050</td>\n",
              "      <td>0.916500</td>\n",
              "      <td>0.915682</td>\n",
              "    </tr>\n",
              "    <tr>\n",
              "      <td>3</td>\n",
              "      <td>0.090600</td>\n",
              "      <td>0.396338</td>\n",
              "      <td>0.915900</td>\n",
              "      <td>0.915638</td>\n",
              "    </tr>\n",
              "  </tbody>\n",
              "</table><p>"
            ]
          },
```

```
          "metadata": {}
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "TrainOutput(global_step=7500,
training_loss=0.18953113848368328, metrics={'train_runtime': 1728.5391,
'train_samples_per_second': 69.423, 'train_steps_per_second': 4.339,
'total_flos': 1.57866633216e+16, 'train_loss': 0.18953113848368328,
'epoch': 3.0})"
            ]
          },
          "metadata": {},
          "execution_count": 16
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [],
      "metadata": {
        "id": "Zml3IipWpiOF"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "eval_results = trainer.evaluate()\n",
        "print(\"Evaluation results:\", eval_results)\n",
        "\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 75
        },
        "id": "Gus2lW5hpif0",
        "outputId": "bb7f8489-06aa-47f1-d543-15a2e09b40b3"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ],
            "text/html": [
              "\n",
              "    <div>\n",
              "      \n",
              "      <progress value='313' max='313' style='width:300px;
height:20px; vertical-align: middle;'></progress>\n",
              "      [313/313 00:37]\n",
              "    </div>\n",
              "    "
            ]
```

```
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Evaluation results: {'eval_loss': 0.30304983258247375,
'eval_accuracy': 0.9165, 'eval_f1': 0.9156821165303444, 'eval_runtime':
38.0592, 'eval_samples_per_second': 262.749, 'eval_steps_per_second':
8.224, 'epoch': 3.0}\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "prediction for test datasets"
      ],
      "metadata": {
        "id": "IqoedQqesITZ"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "test_texts = test_df['clean_review'].tolist()\n",
        "test_encodings = tokenizer(test_texts, padding=\"max_length\",
truncation=True, max_length=256)\n",
        "test_labels = test_df['sentiment'].map(label_map).tolist()\n",
        "from datasets import Dataset\n",
        "\n",
        "test_dataset = Dataset.from_dict({\n",
        "    'input_ids': test_encodings['input_ids'],\n",
        "    'attention_mask': test_encodings['attention_mask'],\n",
        "    'labels': test_labels\n",
        "})\n",
        "predictions_output = trainer.predict(test_dataset)\n",
        "pred_logits = predictions_output.predictions\n",
        "pred_labels = np.argmax(pred_logits, axis=1)\n",
        "\n",
        "# Optionally, print accuracy and F1 on test set\n",
        "from sklearn.metrics import accuracy_score, f1_score\n",
        "\n",
        "acc = accuracy_score(test_labels, pred_labels)\n",
        "f1 = f1_score(test_labels, pred_labels)\n",
        "print(f\"Test Accuracy: {acc:.4f}\")\n",
        "print(f\"Test F1 Score: {f1:.4f}\")\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 53
        },
        "id": "hx9afT67r35a",
        "outputId": "834522b3-cc81-45dd-de87-18f488f342b3"
      },
```

```
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ],
            "text/html": []
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Test Accuracy: 0.9165\n",
            "Test F1 Score: 0.9157\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "prediction for validation\n"
      ],
      "metadata": {
        "id": "ouewVhUMsClw"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "\n",
        "# Predict\n",
        "predictions_output = trainer.predict(val_dataset)\n",
        "logits = predictions_output.predictions\n",
        "pred_labels = np.argmax(logits, axis=-1)\n",
        "true_labels = predictions_output.label_ids\n",
        "\n",
        "\n",
        "print(classification_report(true_labels, pred_labels,
target_names=['negative', 'positive']))\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 178
        },
        "id": "CZldMlRtqXGG",
        "outputId": "79ee16d3-f47f-4bfb-a79f-38af86793845"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
```

```
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ],
            "text/html": []
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "              precision    recall  f1-score   support\n",
            "\n",
            "    negative       0.91      0.93      0.92      5000\n",
            "    positive       0.92      0.91      0.92      5000\n",
            "\n",
            "    accuracy                           0.92     10000\n",
            "   macro avg       0.92      0.92      0.92     10000\n",
            "weighted avg       0.92      0.92      0.92     10000\n",
            "\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "metrics visuals for test\n"
      ],
      "metadata": {
        "id": "Ehj1u0LirWvV"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "# 15. Visualize results\n",
        "metrics = {'Accuracy': acc, 'F1 Score': f1}\n",
        "plt.bar(metrics.keys(), metrics.values(), color=['skyblue',
'lightgreen'])\n",
        "plt.title('Test Set Performance Metrics')\n",
        "plt.ylim(0, 1)\n",
        "plt.show()"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 452
        },
        "id": "afbpsmIasuPF",
        "outputId": "49936de0-e728-4ddf-ad2e-75e20d1be3ab"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
```

"<Figure size 640x480 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAiMAAAGzCAYAAAD9pBdvAAAAOnRFWHRTb2Z0d2FyZQBNYXRw
bG90bGliIHZlcnNpb24zLjEwLjAsIGh0dHBzOi8vbWF0cGxvdGxpYi5vcmcvlHJYcgAAAAlwS
FlzAAAPYQAAD2EBqD+naQAAMfZJREFUeJzt3XtYVHXix/EPkAwIAiIISip411QsTEMrtDA0dT
NrMytBykutlMbPSrcUzRS7me5mmdfKiyLleqGs1LyRrgmipgiuqMz394ePUxO
gjpe+oe/X88zz5Dnfc853hktv5pyZ8TLGGAEAAFjibXsCAADg8kaMAAAAq4gAIAIISip4llQsTEMrtDA0dT
8DLy0ujR4+2PQ1cRogR/OF5eXmd1W3lypXnfawjawjR45o9OjRHu0rNzdXqampatCggfz8/BQRESHu4cq
kbb7xRGRkZ5zSHpUuXevQ/go4dO7o9DqGhobr22ms1a9YsOZ3pDRR599FF99NFHGjFihL
MMaMuXbpc0P1fbk59zfr371/u+ieffNI1Zt++fR7vf9WqVRo9erkQ9erQOHjx4njMFLi4vPpsGf3T/
+Mc/3P795ptvavny5ZozZ47b8s6s9Z7Lh8cjrQVz3pncprTmMeiyxP+aprAoxQvLQp1CuZE9ilvWIlrKJHsKdRPtYWZC8Yq/ZzS
2lfKpo4QdmmFmCoYmz7r0vyYtq8jdQm+GKoiVfNMmfsfyCmQZ0ZGxYZ0ykVjwmmpO8lr/RE9RbnAVzT8nMqNwRPjTOcrvp+NDM9nf7Sc54wA
XlTrPSO1BKCRGWW3DrHfHfLo2iQKDQ9Hx8BrxhJkKFKxxPHpZhTPlHHTJxUjxx2oiM2/KQAb6q5/Wf0Ki/Brz1wHLD6ZxJZ+wwOb6VB0aEoz9dnmwE=
ZYcnqtxdGeHdbKXXUq9sTmZJ/WpVwQd+vPMC+Ro59bm6+ZFNRE7+tk1s
ZUqVc553ydOnJDT6ZSvr6/27t2rkJCQCzDjk44ePSpfX195e1++T9J26dJF7733nj788EPddt
ttruWrVq3Sjh07tdcd+jdd9+96PNwOp06udyYK46A39P1+xsAlxSn06lJkybpcotjZkfrypu/
tInvZZ2OHUllwxelpSBmR0bG75+eSb07TU53D7f70+VEoFbH3DN0H0CE2NrmTpDHF1fj4CSlUCKEKxr
XL9ZdCgJ6yGWmQiXvVgMzKWV6TdlEdGgbDJMYxL6wnmUJSdmQm9jZ6WYdt+5Ws1Ecu4JQ/LtAP1uVd
DoqTJdbdVH7o/JyVLWwuA8+BvV26qKi5XtqK1FxTzkCHu6jPkKGSRPtQJFC9u6ACD2YjIOkZuDO9o9PpTsEz
fn5+Wrfvr2O2vWiv5E5dTVKNSQbzkUDNCHRV3/ttMbCAm+GdnycbVbsJzn+QhTVrZOxi6d+vKomrSmCGT
pLT9WfqdeeOnzeQaNLuNl6Bw13r2zWKKxNCLx/jEAESwSBmsldEyRwN6yfmcAEgAJJ12/fvpbn8Ow4f
9TrXnYovQtZN47zqu8FyZx2Pg4sVufALSp7KLNSx+ZRI6cQwFnnuUkaTANxHbPN1NbauPA0wH79s/Ua8A
oEGqW7euVq1apREjRmjPnjj1lLkSdPXu2jh49qoEDB8rhcOimus7nJycmus
zgrv2OHDmiRx55RDVq1NAbb7yhP/3pT1q4qP4Fuv/12t32PHTtWvr6+GjZsmEpKsLynJK
pLS9W1a1fdeOONeu655zR37lylpaUpICBATz75pO9915oAZNidOVXJysuLj4xUTE+O6r4p
XL9af//xnxwTEKD8/X6+9poSEhL0/ff3bt2m5zmzBDBhgry9vTVs2ADdOnRRIrz33nO6n02
9dVXrjHLly9X9X9+7dVatWLQ0ZMkSRkZHauthGjPvjgAW0ZMkSS9NlX7D7Dhw6Fu8DuROHDFRAQo
HfeeUc9e/bUu+++W+a+V+aSeSee+7dPRkFDVFRUpMDAQJ4cUILFxxtnd6ZcbbC5bJadNFmBmA9v
HKyMiQt7e3Zs+nEtuvLXzpqUE9XW7Qb9BbhD1kxkTIbuqGdcH8HR6erZtq1n6GvTe3eReDhR1Mj2uBV0ekpEvMbyDmdBiGwFQDuD4Hc0k6k7Jl7fX6DFUZW5QLyG6cWBEFq/Bl0zvqGzhW4yr0rFhRdAxE8hLeFMdV4S8RBiHF+xJ6nKepi
FhYVVeCrnp59+Utu2bXXw4EENHDdhQTZs2le7du7Vw4UIdw6KAGJB1K/mxmsdGTvgkPd33F6qsPHtIY8/Ptb9Ua0urOCDhfkWtDmNgRISPJVU5
hYWFWrNmjdatW6f0uf1eOAy5jBpdbswsmjjbu7tzaxpSvv6LBCZhGyRpkwTk7Tau7Rtg/Rk8tZa61mu4bh3uxuWnR4kt6aiSJVT+
e9/V7jvgoICIlkZGSc1vZ+/fZb4+/vbySZ1q7CpMgjf5b4PZ3Ovgx2pXrLSAa
7L8/LyTHBwsNvy396M/M0lISDBNmzUY1BQUFpqCwGzcuNE88sgjkY8b2hr8ODsFqF4JsaFpKHm4PkJg
DDz+4bTt8+HDj4+Njdu7caYwxOeHUaSCQoKMnv37i1zLElm8ODBUGWQPbq8iWd8m8G9V6j1WUG8lBgPpY9+x4Sai
xMSY6OhoU1paaowxsWKFUaSqV+/vjly5IbPlJSUowkM378eNeyn3/2fj3/2fj3/+2fj7+xsvLy/z9ttv
5Zv2rSpzNfo6NGjruOcsmPHWnFDuNwOMzTTz/tWzSnnTzaSzIIYNG4wxxcc
LExMSYevXqmZ9//tlw8LfvoeOgvsz//ubYBQNmsWJcbLy8vk5uaajwIwMI8kUFBS49t+oUSN0bpWYvNKk4+V1l5zvGZvHqyL/
r5syZY4wxZsmJcbLy8vk5uaajwIwMI8kUFBS49t+oUSN0bpWYvNKk4+V1l5zvGZvHqyL/
3kgyO3bsKPfY3t7e5rvvvit33a8f3+TkDZ0nxD7+evTh8UP6M6fDn6...
Dg4WJ07d9a+fftct7i4OBMf48cvyLGvuuoq5eTk6eL777lnNubq4mT6L777lNubq4mT5
6snj17KiIiQtOnT3eNW758uqQaQnc3eNW758uQ4ePKg+ffq4zdHHx0ofts2rVzzfFcbdq0SeHh4Qq
/Xd26ddOsWbMkneyXXmAWy4jMbwjpbBlWvXt3t+IXOsLLdWnn37qtr877rjD7rjD7S/o01m6dKnr66+
3rUsMDBQAwcOVG5urr7//nu8SkpKGfilPvwLbs/1/4fDb/VVlISIiaNGmIAA3XXXXa7lTHNKdiTA+
LZv3+5a5nA4WsL4d/EdJQGoJAUNEI/+LQoMDBMTJTkL2dtt2dbpCTXPtZT1hvz+5kbzCwnTLmnK6NNbqvuOR
ChZa6ROXXzV66QsCBA/rkk09011ElHdP/+/JKVCWZTjTXFlKFJpoJJdaOH/a76bx94Wz9itHRx7xN
6S5I0b948tW/fvtzTfz/6ZYWfGnZ4vF28d+d3DovDSp1wCFGTDtE4OcXV7HfXf/PqO7/3SLJP/3cKJRQL6+g
OJ1OLV68KWfiezPmqmnrS31XDAKQBccSMKJGxLlhsh8tr1wAbcBRhhauPCxhMtacvs5qHXaSeXX1uOXQkdwO
CEhQXfccYfGjBnmjL156SR07dlTPnj1zz0yyO9FTwTDHfkVrZrGGtpk6Q1dqVwOHqFeTAFbAJU+jVrCvW3N

MaOHCgYmJilJiY6PoFXdGpnKCgoHM+viRFR0dr+vTprldnNGrUyO3x2LJli7755psKA+O3F9u
eOv1xNn788Ue1a9euzPJmzZq51v/6uoeK9u3n51dmfsHBwbryyivLXDcTHBzsdj2Q0+nU5MmT
9corr2jHjh0qLS11ratRo0aZY9WtW9ft39WrV5ck1z63bdsmSRVeryGdfCWVMUYjR47UyJEjy
x2zd+9eRUVFVbiPX7vnnnvUt29f7dy5U4sXL9Zzzz1X7rhT30spKSkV7uvQoUOu+3Q6Z/N1Li
goUGFh4WkfC0l6+umnddttt6lx48Zq0aKFunTpor59+572NCdwCjGCSs/pdKpmzZaqaO3duuet
P/Q/Oy8tLCxcu1Jdffqn3339fH33ke6//369+OKL+vLLLxUYGHhe8/Dx8VHLli3VLxcfH
q1OnTpo7d64SExNdf6nOmTNHkZGRbY935dQBgQEKDExscL1TqdnTt31uOPP17u+saNG7v9u
6JnLi6Eivbt4+Pj0XLzq5c+jx8/XiNHjtT999XiNHjtT999+vsWPHKjQ0VN7e3ho6dGi5zxKczT7P5NR+hw
0bpqSkpHLHNGZY8Kz396c//UkqXaovOM+//zzat26dbljzvZ7+UJ+UZ
t27ZN//u//6t//etfmjFjhl566SVNnTq1wvdRAU4hRlDpNQB9//LE6dOhwVr9cr7vuO0l13
3XUaN26c5s2bp3bp3vvvVvdvv/22+vfvf06vvCXCnPqaez9zZ45qjdPIVNqeLBkkXbA6/1qBBAxUVF
Z3x2OeiXr162rx5c5nlmzZbcssPHjzo8ztQWDbssPHjzo8XtzSL98vb799tsKH7P69e
tLkqpUqXJBHld/f3/17Nl8Nlt//jHP9S1a9cK531qbkFBQb/L91J4eLiCgoL07bffnnFsaGio
TlZqaqqKiIt14440aPXo33XWXSktLNXbs2DLrTpw44Xr3yZ9//nMX7Cz/ZxNDh
KSmRdPIlsZLO+h0rP/vss3KvP1m6dKmtkk9c3FJSUp

KCgoI0fvz4csefevmtdPJZDk/mcDbuu
usuZWdn66OPPiqz7uDBgzpx4sQ57/vWW2/V6tWrlZ2d7VpWXFysadOmKTo6+ozXJFwIPj4+Zb
62CxYsOOtrNn7rmmmuuUUxMjCZNmlTm63Dqz7oDVr1lTHjh312muvuaLz1379NT1bw4YNU0ZGRoW
nfSQpLi5ODRo00AsvvvKCioqLTHvdFCfC95e3urZ8+eev/997ev/997VmzZoy6089Hvv373dbHh
G7p+toDT4ZkRVHoJCQkaNGiQMjjzdzlUOTo1tuuUVVqlTRLi1btGDBAk2ePFl33nmn3n3jjjDb3yy
iu6/fbb1aBBAx0+fFjTp09Tp09XUFCQbr31Vkkn/zpt3ry55s+s+fr8aNGys0NFQtWrSo8Hz5s88+q7
Vr16pXr16pc+Pr1q3Tm2++qdDQUA0dOlTSyb9iX3/xFvXt21fXxHON7r77boWhWvnmE2vnzzp1smsS
JOnToJdfflnSyf/ZSNIjjzyo6Pnf67v7z9vx+ixxxx7Te++9p+7du36v36z4j4tTcXGx
NmzYoIULFyo3N/ecnkGQpOHh+ut95S165d9cgjjyjG0NFRvvGGGduzYoXxffffd3eUOz7t276
+mnn1Zqaqaqaqrat2+vDRs2aO7cua5nLz1z7Lz7l7e2tV199VT169FDr1q2VmpqqWrVqadOmTfruu+9cUT
dlyhRdf/31atmypYPYMGKD69esrPz9f9esrVz9S1a1ddddV
VSk1NVVRUlHbv3q0VK1YoKhI77//vqRfvpeefPJJJ3X333apSpYp69Op69OjhipSzNX78eP3rX/9S
QkKCBg4cqGbNmnnPnj1asGCB Pv/8c4WEhKh58+bq4z8LzYL4F66MqTsg9AfnLNhq7L2vFqY94dwGF1b8Hn7jBr1ClZ1Ks3PTKv9N0izWKDp7p5fkV8n3ozlBHqGJHJxA8wB
QcGBbNmbtPZxYkOsdtbHNCB8Poy4iZGdedwC1Eh5QX0jgf/wtSOGSzwPJTk7Dal3VtUGw1MUpA5SVRd8aE...

0zSZmZkaM2aMJ1MDAACV1EV/Nc3KlSs1fvx4vfLKK1q3bp3++c9/asmSJRo7dmyF24wYMUKHD
h1y3Xbt2nWxpwkAACzx6JmRsLAw+fj4KD8/3215fn6+IiMjy91m5MiR6tu3r/r37y9JatmypY
qLizVw4EA9+eST8vYu20MOh0MOh8OTqQEAgErKo2dGfH19FRcXp6ysLNcyp9OprKwsxcfHl7v
NkSNHygSHj4+PJMkY4+l8AQDAJcajZ0YkKT09XSkpKWrTpo3atm2rSZMmqbi4WKmpqZKk5ORk
RUVFKTMzU5LUo0cPTZw4UVdffbXatWunrVu3auTIkerRo4crSgAAwOXL4xjp3bu3CgoKNGrUK
OXl5al169ZatmyZ66LWnTt3uj0TMmLECDkcDo0cPVy+80ee/fAfXDtXJQBHgU0GfUXV7KwxaW
DlzzVZzgQmF3nH6zRPLAXZGzb4RKO+p2qZ9SDpqb1ulfxufD8RsQ7mF+u44jg+JF/qHV8rmrD
8XuftqYM+nPfGewu/f/Kj+2HS/qH/MHoTkqeDf1ECBxAZGVCdrq/0iF2zTavqeqKvZzVuJFsV
dB1yEnWuOmvjrL/yKY/dvJh3X6KxzyFNmP7KxTXc/i6qGTeShNpysWR5cKVnUadwuUsVwUXZf
7Awuc/Np5fuzW1VOmiNqa5XHiDpuf3zfPrzkBz3kdrNyf3Aq/Rve3VNucqurpnbdwFz0SOGSgCT
//81n0wAAAKuIEQAAYBUxAgAArCJGAACAVcQIAACwihgBAABWESMAAMAqYgQAAFhFjAAAAKuI
EQAAYBUxAgAArCJGAACAVcQIAACwihgBAABWESMAAMAqYgQAAFhFjAAAAKuIEQAAYBUxAgAAr
CJGAACAVcQIAACwihgBAABWESMAAMAqYgQAAFhFjAAAAKuIEQAAYBUxAgAArCJGAACAVcQIAA
CwihgBAABWESMAAMAqYgQAAFhFjAAAAKuIEQAAYBUxAgAArCJGAACAVcQIAACwihgBAABWESM
AAMAqYgQAAFhFjAAAAKuIEQAAYBUxAgAArCJGAACAVcQIAACwihgBAABWESMAAMAqYgQAAFhF
jAAAAKuIEQAAYBUxAgAArCJGAACAVcQIAACwihgBAABWESMAAMAqYgQAAFhFjAAAAKuIEQAAY
BUxAgAArCJGAACAVcQIAACwihgBAABWESMAAMCqc4qRKVOmKDDo6Wn5+fmrXrp1Wr1592vEHDx
7U4MGDVatWLTkcDjVu3FhLLy49pwkDAIBLyxWebjB//9//nylLp4kcDj9AAAVFoev7S3d+/nHLDH
ZNWvWLDP+2LFj6ty5s2rWrKmFCxcqqKipKP/74o0JCi7E/AEAQCXncYxMnDhNM2xY8e0du1aqiRp
6tSpWrJkiWbNmqXhw4eXGT9r1iwdOHM2xY8e0du1aJ
SYm/rIDb28lJiYqOzu73G3ee+89xcfHa/DgwYqIiFCLFi00fvx4lZaWVnickpISFRYWul
ClyaMY2bdvn0pLSxUREeG2PCIiQnl5eeVus337di1cuFClpaVaunSpRo4cqRdfffFHPPPNMhcfJ
JzMxUcHCw61anTh1PpgkAACqRi/5qGqfTqZo1a2ratGlKi4tT79699dRTT2nq1KkVbpORkaHQ
OnTIddu1a9fFniYAALDEo2tGwsLC5O5OPjo/z8fLfl+fn5ioyML
fLy8nTs2DH5+vqW2cbhcMjhcHgyNQAAUEl59MyIr6+v4uLilJmZ6VrmdDqVmZmp+zy
GDtm7dKqfT6Vr2ww8/qFatWuXGCAAAuLx4/DBNenq6pk+frjlz5mjTpk0aNGiQiouLlZqaKk
JyRoxRr/EMPPaQDBw5oyJAh+uGHH7RkyRKNHz9egwcPvnAPAwAAVFoef7S3d+/eKigo0KhR
o5SXl6fWrVtr2bJlrotad+7cKS+vU40THR2tjz76SGlpabrmmmsUFRWlIUOG6MknnzxwjwIAAF
o5SXl6fWrVtr2bJlrotad+7cKW+vU40HUNB5Xqyz2g1+4VUn+TqNQwaRRxYqo3uQSDGDZVYC9nlNn+FJJS2vHjbBA0vj/sB7FPLxTJxpRyn0T6Gkz4E
AAqLY9jRJLS0tKUlpZW7rqVK1eWRYfH68vvvtySXsm6QRFGvd4bAA/3hGvVpKyy/SDQx+AM5I3y/8IsVvyvmfkB3O
IEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIA
riBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIA
AKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXEC
AAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVh
EjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAAB
YRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEA
AGAVMQIAAKw6pxiZMmWKoOj5efnp3bt2mn16tUVbjh37lzl7Op06cNhZ7JM+ZAAAUPpamp
fOVnp6ujIwMrVu3Tq3NiaEtAEqa3ta7tmT78V9O0ntFJQthPHRqiHLIBw9lGz0vnZ7GAvmhDkXG1
amqnnz5po6daqqVq2qWbNmVbjhNmVbhNaWmp7r33Xo0ZM0b169c/4zFKSkpUWFjoaUAHonZJ1Kr
jWrt2rRITE3/Zgbe3EhMTlZ2dXe7hBrR7Mc/9tGrWrKkHHnjZHHgrI6Tmz4mPmLF9BAQq1v4EqeTBMAAFOi
HsXIvn37VFpqoaoiCLFlERERysvLvK3ebzssL3ebzzz//5f0j
FCJXHExd3748GH17dtX06dZ9pXxSBgb/O9ZP
dkZGSZ8du2bVNubq5O0k8e+orHHmEHvZ0Ynn2hHmJZ93APTjfzp7YyAgIdcuDRfX70JHZXQ
LnE6nsrKyFB8fX2Z8z806NZXyYn3xWqLTUCKl4mpeWUHHtnJHhGgbucOEmexRlbcRnjbtZnq
tW3bVpMnTsZ9Jnr2h6T9JSw5+uHyX3HS1BsqCUqQaXE3SelY95Q7H5iyXYWoDr8aOqrSeN38e5
gKNGjVKeXl5at26tZbZ5nH6eOhXLQcZAX6S6vLAj8ALS1LdmQlGuS5skRFCMA0i0AbCa1Pr5BCN
wOCQAALlE8hQEAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQI
AAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXE
CAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAV
hEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAA
BYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBE
AAGAVMQIAAKwiRgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwi
RgAAgFXECAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACrzilGpkcrilGpkyZoudaoaPn5+an5+aXavPxr
WOnT5+uG264664rvXSs+2+4Qrd3V0K/e6v21XFPy5s6ZLmqVu9wSxrMwIgcj8CAsJoxOR5q
zxK1euVJ8+brTp3EzgpCmQl5Z20erTp06SZn2YJ0hSTaO3r7uSjfbXKWmOuVxNAsAsS3l2MRLe
UVa1aVbNmzmS3p3/Ny5c/Xv/XXlE/e5ZlHUTnTj4ghvYyPf1NjWxlJyFhaCQAwKXJoxg5
duYY1q5dq8TEK8S2ExF924AJ10XwRaG4vXnExNb6sTzlCXTSXnd8N+lcL7m5XM1cNmMu61UzXdt
eJRjOzbbtt0+lpaWKiHhlEVtZGtYK8aWSZnPl8I4w0IMOd46u1xaG/vVjFfIhNO28HPaqOi
VyeIKj5mVtQ9TvTBnCJe+PL9+JMlBIVZ2A9r7aCFOjIwCAePuBj8hIchPI1MHzRiuMTOnMx9nRTpoi2wd8VgNMQUeQ8YDi8/4yGu
ZedptX3jBU2YMEEeff/yWrVq5q5cSQ37zW4h3dPMjfhWxH8o6+FEmK6C+hPPSP5HVF68
sVq2bJnatGlz7rMFACXtSC5WCXHI9P06SnpylJUVt2rRR27RL2eelVA0jKxVRLaKaUtGcVK18nVgK
kqSnn32WY0aNUsk4XaBaNUrz5s1SnzdsKsMBBABQYGXsC7AgAAKiOPYG3794qKW2

vZsmWui1p37twpb+9fnnB59dVXdezYMd15551u+8nIyNDo0aPPb/YAAKDSO6cLWNPS0pSWllb
uupUrV7r9Ozc391wOAQAALhN8Ng0AALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTEC
AACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVx
AgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAF
YRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAA
AWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gR
AABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsI
kYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABg1TnFyJQpUxQdHS0/Pz+1a9
dOq1evPu34s8PLli21dOnk34BQsWqGnTpvLz+h1PLLi21dOnSc5osAC49HgcI/Pnz1d6eroGDKik
pSXv37i13/KpVq9898MADWr9+KmePXvq22+/+vXr7K4m8KmePXvq22+/Pe/JAWs/JAwCAys/jGJk4caIGDBig1NRU
NW/eXFOnTlTlXVqlU1a9ascsdPnjxZXbp00Zmz2Zpp7Nixuuaaa/Tyyy+/+QBAEDld4Ung
48dO6a1a9dqxIgRrmXe3t5KTExUUqlUpUL15c4XFK
ckSYWFhZ5M96wcLTp8wfcJXEoKC31tT+GCOFp41PYT+//bxtjTjvOz0tOmzPI6O/4GnT
fSktLFRER4bY8IiJCCmzZtKneVeJzMzEyNGyPOTNmEyEyNGTOmzPI6dep4M0Ml0AF0DZn0QAl5rh
Gn5R93/48GEFBwdXuN6jGPm9jKqvKfQMHDqhqhGp4527dqlo
KAg29MBcJHws375MMbo8OHDql+vtvy/Px8RUZGlrlGxSZrtNZGSkR+Mlye
FwyOFwuC0LCQnxZKqgvgFBVwG+Fm/PJzuGZFTPLqA1dfXV3FxccrKUnx
8fLnbxMfHu42XpC4c4HgAAXF48Pk2Tnp6ulJsULvJycrKi
oqKUmZkpSRoyZIIgSEhL04osvqlu3bnr77be1a9cuvmLeEwAAUCl5HCC9e/fW9u3bNWrUKJWUlKh169Yq
by8PLVu3VrLli1zzXS8+c/8oRL+/btNW/661//qkaNGuk//7P/6KqrrrqQ9wMAAFSSXmsXbXuTuC
Q4HA5lZGGSUOH4NLCzzp6fU2AAAAFxEgTQMAAAsVhEgAAAsIoYAQAAVhEjAADAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFX
AKmIEp5WdbN0flXXT+QIqkVCX+6WVy++t9bydKqlFDXjWbI5ZjyOdbHXAKmIEAABYRYwAAACriBEAAGAVMQIAAKwiRgAAgFXEC
AACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsIkYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABgFTECAACsI
kYAAIBVxAgAALCKGAEAAFYRIwAAwCpiBAAAWEWMAAAAq4gRAABg1TnFyJQpUxQdHS0/Pz+1a9dOq1evPu34s8PLli21dOnk34BQsWqGnTpvLz+h1PLLi21dOnSc5osAC49HgcI/Pnz1d6eroGDKik
pSXv37i13/KpVq9898MADWr9+KmePXvq22+/+vXr7K4m8KmePXvq22+/+vXr7K4m8KmePXvq22+/+vXr7K4m8KmePXvq22+/+vXr7K4m8Km

```
bm18fX1NWFiY6dWrl2vcf//7X/Poo4+aWrVqGV9fX9OwYUMza9Ys1/pFixaZhg0bGn9/f9O9e
3czbdq0M8aIMcYsX77cNGvWzDgcDtOqVSuzcuXKMhfU5ebmmjvuuMMEBQWZqlWrmjZt2pivvv
rKbT8zZ840kszq1avP8VECLh1nipEVK1YYSWVuvw6D35o2bZrp1KmTCQ8PN76+vqZu3bqmX79
+Jjc31zWmtLTUPPPMM6ZevXqmSpUqm7dumb8+PGu9d98843p1KmTCQ8PN76fPzM6GhoWbAgAHm8OHD
Z5z3nj17TJHJysgkLCzMOh8PUr1/fDBgwwBw6dMijxxW/Hy9jfnPyHrgMjB07VgsWLNA333xje
yoAcNnjmhFcVoqKivTtt9/q5Zdf/sMPP2x7OgAAESO4zKSlpSkuLk4dO3U/fff1sMPP2x7OgAAESO4
YBAABW8cwIAACwihgBAABWESMAAAMAqYgQAAFhFjAAAAKuIEQAAYBUxAgArCJGAAABWESMAAAMAqYgQ
Eho0uaKIAAAAASUVORK5CYII=\n"
          },
          "metadata": {}
        }
      ]
    }
  ]
}
```