Multi-modal RAG with LangChain

SetUp

Install the dependencies you need to run the notebook.

```
# for linux
# %sudo apt-get install poppler-utils tesseract-ocr libmagic-dev

# for mac
# %brew install poppler tesseract libmagic

%pip install -Uq "unstructured[all-docs]" pillow lxml pillow
%pip install -Uq chromadb tiktoken
%pip install -Uq langchain langchain-community langchain-openai
langchain-groq
%pip install -Uq python_dotenv
```

```
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
```

```
import os

# keys for the services we will use

os.environ["OPENAI_API_KEY"] = "sk-..."
os.environ["GROQ_API_KEY"] = "sk-..."
os.environ["LANGCHAIN_API_KEY"] = "sk-..."
os.environ["LANGCHAIN_TRACING_V2"] = "true"
```

```
True
```

Extract the data

Extract the elements of the PDF that we will be able to use in the retrieval process. These elements can be: Text, Images, Tables, etc.

Partition PDF tables, text, and images

```
from unstructured.partition.pdf import partition_pdf

output_path = "./content/"
file_path = output_path + 'attention.pdf'

# Reference: https://docs.unstructured.io/open-source/core-
functionality/chunking
```

```python
chunks = partition_pdf(
    filename=file_path,
    infer_table_structure=True,            # extract tables
    strategy="hi_res",                     # mandatory to infer tables

    extract_image_block_types=["Image"],   # Add 'Table' to list to
extract image of tables
    # image_output_dir_path=output_path,   # if None, images and
tables will saved in base64

    extract_image_block_to_payload=True,   # if true, will extract
base64 for API usage

    chunking_strategy="by_title",          # or 'basic'
    max_characters=10000,                  # defaults to 500
    combine_text_under_n_chars=2000,       # defaults to 0
    new_after_n_chars=6000,

    # extract_images_in_pdf=True,          # deprecated
)
```

```
/opt/homebrew/Caskroom/miniconda/base/envs/mm-rag/lib/python3.12/site-
packages/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please
update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Some weights of the model checkpoint at microsoft/table-transformer-
structure-recognition were not used when initializing
TableTransformerForObjectDetection:
['model.backbone.conv_encoder.model.layer2.0.downsample.1.num_batches_
tracked',
'model.backbone.conv_encoder.model.layer3.0.downsample.1.num_batches_t
racked',
'model.backbone.conv_encoder.model.layer4.0.downsample.1.num_batches_t
racked']
- This IS expected if you are initializing
TableTransformerForObjectDetection from the checkpoint of a model
trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TableTransformerForObjectDetection from the checkpoint of a model that
you expect to be exactly identical (initializing a
BertForSequenceClassification model from a
BertForSequenceClassification model).
```

```python
# We get 2 types of elements from the partition_pdf function
set([str(type(el)) for el in chunks])
```

```
{"<class 'unstructured.documents.elements.CompositeElement'>",
 "<class 'unstructured.documents.elements.Table'>"}
```

```
# Each CompositeElement containes a bunch of related elements.
# This makes it easy to use these elements together in a RAG pipeline.

chunks[3].metadata.orig_elements
```

```
[<unstructured.documents.elements.Title at 0x3316571a0>,
 <unstructured.documents.elements.NarrativeText at 0x3316574d0>,
 <unstructured.documents.elements.Footer at 0x331657a40>,
 <unstructured.documents.elements.Image at 0x331657710>,
 <unstructured.documents.elements.Image at 0x331657650>,
 <unstructured.documents.elements.NarrativeText at 0x17bed0e90>,
 <unstructured.documents.elements.NarrativeText at 0x17bed0830>,
 <unstructured.documents.elements.Title at 0x17bed27e0>,
 <unstructured.documents.elements.NarrativeText at 0x17bed03b0>,
 <unstructured.documents.elements.NarrativeText at 0x17bed20c0>,
 <unstructured.documents.elements.Formula at 0x17bed0ef0>,
 <unstructured.documents.elements.NarrativeText at 0x17bed3d40>,
 <unstructured.documents.elements.NarrativeText at 0x17bed3710>]
```

```
# This is what an extracted image looks like.
# It contains the base64 representation only because we set the param
extract_image_block_to_payload=True

elements = chunks[3].metadata.orig_elements
chunk_images = [el for el in elements if 'Image' in str(type(el))]
chunk_images[0].to_dict()
```


```
{'type': 'Image',
 'element_id': 'ceba98e3-85c8-4d31-810f-90ba503ed344',
 'text': 'Scaled Dot-Product Attention  Multi-Head Attention ',
 'metadata': {'detection_class_prob': 0.8599750995635986,
  'coordinates': {'points': ((410.50833333333327, 197.0675599999999),
    (410.50833333333327, 669.311279296875),
    (1311.755126953125, 669.311279296875),
    (1311.755126953125, 197.0675599999999)),
   'system': 'PixelSpace',
   'layout_width': 1700,
   'layout_height': 2200},
  'last_modified': '2024-10-09T21:49:12',
  'filetype': 'PPM',
  'languages': ['eng'],
  'page_number': 4,
  'image_base64':
'/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UH
RofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/
2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMj
IyMjIyMjIyMjIyMjIyMjL/wAARCAHYA4UDASIAAhEBAxEB/
```

8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fA
kM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1d
nd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NX
W19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFB
gcICQoL/
8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvA
VYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0d
XZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tP
U1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD0KHxFq3ifxxrmhaVqM
GmW2iiJZHMAlluJHBJwGOFRcY6ZOeorS8O3/
iQeKdV0nxA1nJFDbwy2U9rEYxKpZw5YFjhuFBGcDt1rnfGfw4vr7XT4v8F6t/
Z2v7cSLn91c4GMHqAcAA5BBwM461N4F+JFxrI1XS/
EenGw8QaPE0lzCo4kRerL6duMkfMCDg8AHo9Fec+GdKj8feAv7Z1WeQ6jqqSvFOkjf6F8z
KgiGRt24HI5Jzk81d1nSb6z8KaFaar4r2RWdzF/
aFzIpVr9AT+64OctwMDJb36EA7k5xx1rhfC3iHW734jeKdB1S5tpoNMS3MHkQeUP3i7ucs
xzggde1ZuiXzWnxpm0qxtrix0q60QXX2SRdimQS7RIqZ+TI4IIU+oo8MyLD8aviFK2dqQW
THHoIRQB6ZRXj+lQ6v4/8DNq0mmGW/
1ATPa3q6k0RtSHZUEaagWqeJNP0jwZ4V1m8EWr6vdNb313bS/
M0MZy21nA78R9C0/Stf5gQxXON3X5upzzmvXKAPMdT8SeKoUzZGL13g3X5upnyCPUVfr5y8Ha/
8nOaL/2Bm/8Aa1en6r9n/si9+17fs3kSebu6bNp3Z/
DNAEei6xY+INDtdV02XzrS6jEkT4xx7jsR0UeG/
qnh34H2iW8720mq699it7kdYYnA3sueM5VFx9TntXovD0UeG/
B83iDw7LNY6npeyUSLKzfaE3AMsuSfMyDnLZOR1oA7i/
1qw0y90+yTZgtzqEpitohyzkKWJx6ADk+49a0K8b8URWXiDx/
8MtTmtcHVLeaSVSzA7MwTzwbwwbPIOQelQ67rHjTwz4X/
4SP7VdV7pWrW0MKzXNobR7dwhxkxnuvSSn42ahb6r8H+Hft9ozF9YY5/kSyoQk3GPxX/
GtTU9E8Y+LkPTX4n6f3vPjWRjHdRMwQrIpPzGNTvVn4xXDLdt42sWLkvPaw69cQRxyuWCIp4UZ6AYoA
9NorxDwx4svfDWvyd3h6Do7GuXdKzaqbOZeLiqczpUKUYoRAymKS91fCXGBqHEsmMDGsj7sZeZwExuUUaeUIzFydDNyoClQ
D+JVVh8p4yc+lAFqw8Ra0/
xjvjvDV1cWz6bFpKig2MGiKkNxcXyXpFvJ0yLHwQvgyouS8vzHIJDWKnt42Pvb0xNR1dcxMQRgWxq6nfMdJfQdv
c/Q1oV5Nrmjafd/
tB6Wk9uHW40d5JfmYbmDMsOWOwYybK4CFfHmc3dg3RSUA5YcmgcfBgBYHLbnMTaDBSuqsvTVsxXSQ6rtKsoQA2
i/ht46l8ZW+rX1ubW/sbtlMDLtZYWI9cZB/
wB3PepviBqcKxadoUkd1Kmoz7rtLW2Pp0c1wmq+ILTw98ZNL8qSWtvq
FtpusRiw1E3dhNbKH4CPmRVB6L0zgK3rQB69retVdSmHWhvJvdu59AB3JPAHqn
a1qw8P6TNqozCK3ixk92JOAoHck8CuA+PlrDN8MbieSaM2ePMNLBPEY2JPyksAT+WR+NU/
jPoWm2fw
tvItgnl6jCU+djgscN1PcAUAet0V594tu30XVPDPhfR7dkj1a4neZEuDGXSJNxQScldxK5

I5wCO9Rad4d8Saf49s9S06zi07RJIjHqFl9uMqu3O2RVxgNkjOOuPc0AejUyWWOCJ5ZXWO
NFLO7HAUDkknsK8l8MaLHrfxE8e6VqN9qdxp1vJahbd76XBDK5wW3b8DnAzjnkHAxX8NB7
z4Y+PtFvppby20m7vra2M8hZhGiZQE98EZoA9SstUj1/
w+NQ0eddtzE5tZpE+XPIViPTIzj0p2gw6rb6Haxa5dQXWpqmJ5oE2o5yeg47Y7D6CuF8Ba
Paj4N2k8DT21xPpxZ57ed0fI3EEEHj8KwdL8Xaj4e/
Zrt9cimeXUmEscc0pLkO9y67yTnJAJPPcCgD2mivPvE3haK08BXOoaXd3UOtWFobuLUVmY
yyui7zvJPzhsEYORz0rlPEmt3viDw/
8ADfW0vLuyudR1O2huBbzMsZO7k+WflPzDIyD+IoA9soryrUdPTwl8X/
CY0q4vFi1lLqO+jmupJRMUQFWO8nnJ/
SvVaACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAC
iiigAooooAKKKKACiiigBsgZo2VG2MQQGxnB9a8o0z4i61Y/
FqXwrrrW0mmyzNa2l2kPlkzBEcA8kdHC/VhXrNeL+NfDUuu+H/
GN9Y7l1TSNd+22rp975bW3LAfgM/VRQB7DeXcFhZT3lzII7e3jaWVz0VVGSfyFeY/
Djxv4m8Y+K9ZtdS+zWVpYbHW2WD95hySqsxPBBAHPHWrlp4li+Ivh3w5Yw7T/aY8/VEX/
lnFCR5iH0DybF91Y1nfDgA/
Fz4igjI8+Pj8XoA9aqvfLdvZSrYzQw3JX5JJojIinPdQRSfzrzv4eIJPGXxB0+QtJZxR
wOxZEUhyQAeg9hUnweklfR/E0ck0siQa/dRRYYA8aPw08VXfiH4eQ6Q6
rlxCsheZZpZAojREVj+qmFBPYAaPw08VXfiH4eQ6Q6/
2YdRSeMOqWt7KoJPDKXKn8CAfwrpfDOk20PwThuLV7i4uM2mhuL4i0uZdHEzTW87o+9YyQQ0A7z
RYtSg0a0i1i4lVi4hudRWMC4mhTajt3IH/
6voKs3Uc0ts6W6W06DVXYAtpBUh8fCQ0sZLvAY8F5NfH+0mRv1PvnV6OzkgzYjk7mj+1iLCn2WSw
94v0fxS1wulSzyG3CmTzbaSLG70Mb1Geh6UAcr8LPFpiHxaNWuWWRN84dXkBaIKw6d2FO1Re+eWoDI
sSWPqOKZ4v8T+JdG+Inh7LZ0kz7Fh8vZdZjQmU6PT4SuR3/idXIUa63/2wsg3SsGUJXDWhfT0f66eq/
AGvJvF5bSSSwWwOhWRQBj2qXxUmtp8Y/
Af9tJvEvkVFPrwHcvAsj2TW3gGhn7h0nNuGKgbrRSd3MzCEH9bB3GfUC6U3XdV/
AMJ3pvg7gpUMjaQ+5XIYjp8ZcpG2Y9hvE1oWdFJWvNmJQ1bGTnvRSoCNumAg8H/
qvT4E6ulbBU5yc3MzO1o08+WdYUqJ1d17H4vnL5JUVB7P1bt0JOj0MfXtcxxeGlVJaMgBjJnRbN6s
ElsAh1QQEb2PyEgkZz6k0Aet0V53rl9P/
AMJ3pvg7T7My2MOmfy2wujCJSyPBUtySo+Y7e+RnpT/
AAx4d8RaTAZdJ58Ss4xUWMmzLPX1ZkJgcfEYdSRtEkAbnqOuvBD+DT9DUJkpxTVkFZNGixMDq
3QySyN0UD/
AD0715H4A8OxeUwaxqGp3c7HQnjt47uFd7FuZhPZp0kDGg7MhC1e7iDA4z3FtO6uh8rM6frSOpkLRrdpUyFcY0p89jVxCWJd7JYLKtyGJhEXsOybdYJwjqwOu4Fm2u0KMwXI6mGFVpBqQvL7rQYnW5EMYe7nTlMYUEBsVaUeAbBcT22xxjq1nWRD8Tdye1/
bpVHWfEd7pfwx8DWlrdSQXKmR4Sp5tkP7yW7JJPVH4YZ8g0o2Ft5OxNYzm3o0mHOXEfD8pG6tZQd3O8xEsFsMtN2XwydKJYdk9Cm6IC
wCET8Kt4m8OPJZX+lyRu4WRitzGWCssoJ+fqDk88daz/EMs2qfFLwI0F/
qNpb6raYywx3LhR+4J4UnCnBIAB7jB5oA9bory7BCd5xbKOT4LuZJ8EaqYc5Nw0fSxNcuxdBHXbEs37yE4Gg3A6NxhKbcTdNJhUH
lEm3cC5J5APfv9K9RoAKKKKACiiigAooooAKKKKAOK0fTvFvh241LTNMsNPubK7vZLbS2lcNkbbW2
hlgV3Lbc7CpGSTjjGep6BvhzwbdxeLdZ8Wa8beBRvUbhmWshFfL4tLpUM8WV5XVJ0Jdbm/
oA8u8P8AhDx34L4d3qyvr0PHxL9JQ8FR0dWLZWNkPNPPM8WjXgh3k6kmvBPB9Bjy8vi9W0JWCh
B2vXw8NXmjX8FzfaNdcyJqDFUuWbBLHaDtIOcADYwK9AooA89Xwv4wmv4s/

4WVZeKpJtIdW0/7FdRKZAIV37/k4+c+529+BVjw/4Z1/T/
iL4h8Q3semfY9XSFPLhuZGeMRqFHBjAOceox713VFAHl+g+EfHPgma50vw5eaLdaBLM0tu
uo+aJLUMclQF+8Px5PPy5NafirwBf61o+mS2mr7fEWmXX2yG9nUhJJDgspUZ2pwoAGcBQO
ea72igDzPxB4X8b+K5fD1xeNoVjLpV/
HeFYpJZVcpznlR3H3fQn5u1elRhxGokZWfA3FRgE98DJx+dOooA801Hwl4tn+K1r4xtodE
MNram1S2kvpVZ1If5iwhIBy/TB6Vqa34e8V+LrR9M1S/
sNI0mbi5j05nnnmTunmOqhQe+FP5E129FAHIeI/h7pet+BY/
C9tmxgtgjWckYyYHXo3Xnqc9zk9+ag1PQ/Enijw2vh/
WhYW0Muxb68tZ2dpkUgkIhQbS2OSSduTjNdtRQBwvjDwjqd5q/
hXVvDq2Ql0KRwLa5dkR4nVVwGAJGAvp39sV2lqLgWsf2sxG4x+88oELn0Gecf54qaigDhf
il4V1vxp4aGi6T/
Z8aPKksk106Vr38OkW0OpRW0dzFGqMLaZpEOABkFlU9c8Y/
GtCigDhvG3hjXte8c8BVRs9ucj6V1WiR38OkW0OpRW0dzFGqMLaZpEOABkFlU9c8Y/
wmWgwQfaBZanaTJdWlwmXEUy/
lle3bsccYrqqKAOL1nQNb8Y6XbaPr0NhaWPmxyXtbh5Tc7CG2KCi7FJAJJJIxjnrVTWvvB
+t2fjtfF/hWayNyaNxNbi2vrG9ZkjmQYwysoOGGF7dvciu/
ooA8/0zwp4nSeLb650qOKawWzkt4RJIVXcGIUnbn7o+Y+p+WmaP4V8V+G/
E+vNpU+kvpOsXXpluDIZrd35YBAMN7fMOn4V6HRQB5noHwyu4vqvhDXLy2eeSKW3
DFoyWDKWJwCQyg4AHcVa8PaN8Rre0i0XWNV0c6dAoRr6ASm7mQcYwcKrEfxZOO+M807ZzKi
gDhfEXhW3+IOjz+wyfZbV7OeC7ldPkJJJDAqrZxk
BjUHczAk57Ac4qH4m+/Lzxp4P8A0Pl0/E9+ZE7dfOeZ+ntF4ahchMDbjrgrtv8vSWXDy
MxqwUHMYBznk9veq3hXwTrWnad4shTXnad4tsNWbT1j164uLhXtZ3kMZlUhkXzpnrn8K9DooA4Lwp
oPizRfAp0Ap0C7j0d5Le3e3t2SeQCTOcMx2fKAD0AOfUdDBoHw8uR8J28D+ IjalQjqlxZys/
JkaRWwyLgqSOc4r0SigDhk0PxYdh0PxZP4Nha8l04BrfF7JqGiT0zCdDoHhw8uR8J28D+/
rCvH3sZ5z2qr4dW3dW3hOw8PR2EdloF3gzmRcAdDhfEvhnxq/
l9F+VG69znr2r0igDhfEvhnxq/
jrwrzrtr4prxpH4m7gVDhfEvhnxq/
jrwzrtrHpi22zM8KjulupuDH8kjYUna4ufECSL5L1T8Yjp2fsFoUCEqJeJRNewqD8/V7DVtYjGa9JA7lT4vRJnR9Q/G5+QJP
4xMdWJXM5zJcAAp5ZzzXeUAKRgTYvL5uAD5LL6XaPFBkCKxuXnqSTAxA9L5J1OSmDm28JBE
2RprNWQSWsrOrgsWz8yLjr710V01wls7WkUUs4HyJLIY1J92CsR+RqaigDzv4v4Y+D/
ABF4N/
tSDVf7LlhvrprzLW4kLIxHK7WjAI6c5pni3wj4n1v4gaDr9kmLa6O5KxzXcgeYE85xEQ
vA969HooA898R+EPEa+L7bxh4VurCHU2thbX1neM5hnUc8MBnI47DoDxyDqHQfEOs6Hqia
5qNrBf3llLaQR6eH8m2DjBbLfM7E454wBgYSeuooA8gu/AHjXU/
havhK5m0SBrMRRbtC0jeeEYEbiQNnHoDk+lb3iLwf4h1LU9C8Vadc6dbeJtORo5wbWLm2nj
bOU3Y3cZPOOc544r0GigDzm68LecLuOCfTGB1r0ViwRigBbHAJwCfrS0UAcr4V0nXrLWdc1HXU0
4y6hMrxPaXDuY4k0+Y5zyWPFZnxS8G6r40sKNIBd2l0txxHd3E7o0ZAPChUb

OeD1HQV3tFAHAeLvC/
iTxJoOgTg6bHr2k38N6UEzmCZk64bYCMnBxjjpnvTvGHhjxFr9t4e1G3bTl1jSb9bwwM7i
Fl7oHwSTwOcDPPA6V3tFAHnvijwd4j1LU9H8UaPfafa+JbBGikjcObaaJiT5ZPXjJGcDOc
/Lxjd0e08Uzyi+8RTacs0KN9nsrAuIt5GN0jtyTjgADAyT8xxjpaKAOB8D+F/
EfhbTNfguY9Klmv76a+g8q6k2hpAo2MTFwBjqAfpWZpXw41mH4N33gm+nsFuXD+RPBK7ox
MnmDdlARyAOM16jRQBwV9oPi3VfhpceH7mPR476a0W0BW4k2KAAC5bYSSf7oAx6nPDb7wB
c698NNM8OanNBaalpscP2a7tXaRUkiQKr8qp55yO2etd/
RQBxWr6D4g8XaFFoWux2FraSNGb+a0neQzqjBtqKUXYGIGSScDjnrUOu+FdbvPiJ4c17T4
9NXT9Gjlj8qW4dHcSIVOAIyBjPHPPtXd0UAcM3hnXW+LieKtunf2cth9g8v7Q/
m7d5bfjy8Z56Z/
Gu5oooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoooo
AKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKA
CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigA
ooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDyb4yfEWXwde6BZ2D7rj7St
5dRKQC0CnGw+gc7uf8AYr1O0uob6zgu7aQSQTxrLG69GVhkEfga+dfjv4YNv4g0/
Wbu+knn1O5aERKNscEKBAqrnJzyxJ6ZPAHf3Twl4ffwt4eh0Y3sl5Dasy28kow4iJJVW9S
ucZGBgDgUAblFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA
UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAF
FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABR
RRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUU
UUAFFFFABRRRQAUUVFc3MNnaTXVxII4IUaSR26KoGST+AoAlorwW28Y/
Ef4l6ldT+EpoNF0OCQxpLOq5c8HDHaxLY5woAGcEnqb//AAjfxk/6Hew/
75/+1UBc9rorxT/hG/jJ/wBDvYf98/8A2qj/AIRv4yf9DvYf98//AGqgVyP9oj/mU/
8Ar7l/9p17fXz3r3ww+JHif7N/bPiJbTbv7KaHcWXYTjJ+WMegrZ/
4Rv4yf9DvYf8AfP8A9qoC57XXRXin/AAjfxk/6Hew/75/+1Uf8I38ZP+h3sP8AvnP8Avn/
7VQFz2uivEpPD3xnijaSPxnYSuvKijaSPxnYSOOyE2j5vbmLH51u/
DD4iarruq33hXxVAtvr1lGZfMjAUSqCAcqOAQGU/KcEMEHGQ6A9VhhYXKxPHYsyAvHRrRRQAUUUUAFFFFABRRRQA
UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAF
FFFABRRRQAUUUUAFFFFABRRRQAV5N8ZviJN4PuNBs9PkzdG5W8uIg2N0CHGw+gc55/2DXr
NfOfx78K/Zr628RXV9LPc3939nSIDbHBAq/Ko6ktnJJzjLHigD6Gs7uG/
soLy2kEkFxGskbjoyMMgigfMggBAAMA0FiZCCJD4aUlJmGgUMEA1E5EOWbv+UT0sUnBsdTpbQmrj0yl7qcJGS1i13bnXk/
soLy2kEkFxGssbjoysMg/
kamrD8JeHm8K+HodF+2yXkFs7i3klHzrESSqse5XOMjAwBwK3KACiiigAooooAKKKKACii
igAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooo
oAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACue8enHw78S/
9gu5/9FNXQ1zvj7/knfiX/sF3P/
otqAOM+CigfC7TiAATJMT7u2v9mMEOBnLBGfn04Q1xXwV/5Jbpv/
XSb/0a1bfif/kOeE/+wq3/APX+wq3/
AKTT0EnSOwRGc9FBJxVfTr+Hu6HU9Ntr+Ftr+3DiG4jWVA67WwRkZHYHY1LcMUtpWU4ZJB/
CvP8AStW17WG8I2q6q8q8Sj3svqsRRl2YeVyuVIBy5HTGCeM4wAeiUVxdd6h
c6hp0GkKrOPtq6FadadplO8XaL9fcHDtfhHPzl55T1mBmqmlDRS3/
aieKbyTWvJ84Wn2PLdHVCqoU7WkbWa3bQurTzAD2uvBbt9sRSS6qN/
B0Q9+XPS3xGuiY+R1G6OJjewlZ9tWgW0SC+C0Bl0hvRYybQ0nO2/XUa6ntB5i4P9m9AhJ4BXfW2tqHC1Usxt3+tPsWhv
oxdbTNFGQCzmXddY8bJr7h8lSE7mnE4QXlllakrNQQHY2SBe+lwLD3gE8XxNL1dnC/7jymZWNGs3q3slyM0QLf4UM2zmrT
1TbY4zbHPv/ozV6vptlLY2xjnvri9l/zDeMZGrPe3eKyQpGCAiu/+6JVFDuBfD8pxA5ac/xd1Pu8ftw
+kzUAj3aiiigoKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiii
gAooooAKKKKACiiigAooooAKKKKACiiigAoorK8SeILLwt4evNa1At9mtU3FUGWdjwqr7k
kEkDkk9NAGrRXz/8Xr+z/0tD2Kn5+0dvmT7bS0pHwA76KFdHBhyiY1FO4xe36vmQM6mR/TfPgmx4mSKjyBHi5Sq5J2WVgH/
SB+NAGrRXz/D8XfiVrVrDUh+3+f3/+5KjvzmL3ZxD3Cwvle2ZWhFLeEL8z+eXMNaAARk6dzKTt1nGPRxlOTfxwvRl8kb04nmBlE2
8A8dp2YHvlFeB/8LJ+tb+3/C8ztHV0YaImP4/Pu4i6kGdV7/XOX5yqFg4iZIXTxpBMEb0o+sVAiUy8WTUvCAW3+Dqve/
wDjtHK+wHvlleK/8LJ+RB/8I3/0ll/8ACPt/nj/9pj16KV0ugcC/n6OtH/bqCJ+yWwqiZ5mb0cb7T
wDjtHK+wHvlFeB/8LJ+Lfwh/c/9l0V/4Ce8/4QtFWs8NkAFOo4iXB5EuJMVgkNqa9KeJuK4nN3XK/
wDjtHK+wHvlleK/8LJ+tGf8AICO0D/r+P/oNZv/Cyr9A/wDjtHK+wHvlleFgxB7xJkWTZUTSOYr6UbTpRmckuTqtRRMBjEDPRr0g/SCJ
wDjtHK+wHvlleK/8LJ+tG

8SfHNraW+qeG4Y0tZTLH9mjKktjHOXNHK+wH1FRXgf8Awsn4vf8AQq6f/wCA7/8Ax2j/
AIWT8Xv+hV0//wAB3/8AjtHK+wHvlFeB/wDCyfi9/wBCrp//AIDv/wDHaP8AhZPxe/
6FXT//AAHf/wCO0cr7Ae+UV4H/AMLJ+L3/AEKun/8AgO//AMdo/wCFk/F7//oVdP/
8AAd//AI7RyvsB75RXgf8Awsn4vf8AQq6f/wCA7/8Ax2uq+HPxXufEutzeG/
EWmrp2uRqzKIwVSQDkrtYkhsc9SCATx3GmgPUqKKKQBRRRQAUUUUAFFFFABRRRQAUUUUAF
FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABR
RRQAUUUUAFFFFABRRRQAUUUUAFc74+/wCSd+Jf+wXc/
wDotq6KsrxPpsuseFNY0yAgTXdlNBGW6bmQgZ/E0AcB8Ff+SW6b/
wBdJv8A0a1dH4k0rUNQk0m50x7UXFheG423JYIwMUkePlGf48/hXmvwi8c6Po3hx/
DWu3Uel39hPIu27PlhgWJIJPAYMSCD7V6N/wAJ54R/6GbSP/
AxP8aCRY18WSuI7pNFEDfLIYnlLAHrjIxmqmheFbnSrjw9JLcRONM0l7CQLn53Ji04e37s
/mKtf8J54R/6GbSP/AxP8aP+E88I/wDQzaR/4GJ/
jQBLf+H11HXZbq4KNZzaZJYSxc7mDsSfwxkVmppni5NEGiLd6csYj8sYj8hdVSRxOsfQERbceZ
t/i34zzjtV3/hPPCP/AEM2kf8AgYn+NH/CeeEf+hm0j/wMT/
GgCvFpGu6Rq+pT6Wun3Fc8Y333s8iTRMkax8kI3mL8gOCV5Lc81WsfCep29tPb3N7bz+Zq
8Wp+cFKliGVpFK4wOV+Xk8Hnpzo/8J54R/6GbSP/AAMT/Gj/AITzwj/
0M2kf+Bif40AdDXjq/wDJ1Nr/ANex/wDSZq79/H/
hCONnbxNpRCjJ23SMfyBya868Czjx18eL7xTp8ci6Xp8ci6Xp8BQSuuPMJTy1+mfmYey80Aj3yii
igoKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKK
KACiiigAooooAKKKKACiiigAryz9oEkfDJgCRm9iB9/vV6nXln7QP/JMj/
wBfsX8moAd4YUL4T0YKAALGHgf7grVrL8M/8ipo/wD14w/
+gLWpXathhVHVtYsNCsDe6lcCC2DBS5VmwScDgAmr1cx46iSfSLSLCGVQ0cmqWiMp7gygEUm
7IDpwQQCDkHoRVGz1iw1C+vbO0uBLcWTBLhVU4jY54JxxnjPHtmexXDmSfRS1q
Q33pNoHlfiysn4mqWg2V7o+sRLHPqQsLaRy7YV52MzMSfTcT+FLm2A7aoluYHunCBVpYVjT7S/
HxE1V21NmjW2t3aPyFG5C021M9sevU5o5gOg0nWdP1vMKiVwwsQDRe6xY
adeWVpd3jYJLsMZ6DjqOtcb4D4D/4lgsAXxb6tbuyr6TxMQfrxKBwjqekJwOp8mYf0r0GvP4P8Ak5b
QP+vOT/tn6gwr4lqnVRXrk
0TNWdX4QPfKKK5hBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA
UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHJ+JPhp4S8V3n23VdJR7vGDPE7RO3+9tI3dMZOaw/+FE+Af+gZcf8AgXJ/jXpFFFAHm/wDwonwD/0BAy4/8C5P8P8AhRPgH/oGXH/gXJ/
jXpFFFAHzR8WfCngBngbwNd6HTPPNOJ7uL7U5JtlOCvJ4LHOD/smvSrf4I/D+5to5ov7s9RN/
Dy7tYrm3sJpIZkEbreSeSEMpGQQRz6V518eu6PDt1SLOpQ49QIpT9CmOoPqoQCFgE8liWZj/mAzYIN
2yce9e6+D9CufDPPhbu0W4W1aW27wWKIVCSSgHBz5sPzPy+iRLnQrkKdOZ7W3SdD0i9K0XcnLn
9Ay4/8C5P8P8aP+E88I/wDQzaR/4GJ/+jQBLf+H11HXZbq4KNZzaZJYSxc7mDsSfwxkVmppni5NEGiLd6csYj8sYj8hdVSRxOsfQERbceZ
ArwCrAnSp2A7G7lwfyau50jRdN0DTo7DSrOO1tUJKxRjjJOSeep/P8Ay5TtqaoUUUVkvFnw/
vrCxxXfexbiCPON/RatBV52YX1zAj1YmoPkr3kTC0FL1K2ZV3kXlHBZ2WV23TKFGmL7BLz2V/
6AtaleP6N8QdaT/KqOwlcl8Ptdi24tN1xMT/hwhk2h8UZgdmbWQ20uZCaDS/+zyjsvO5zhXvuc
AbrqVSNtxnqNZHiHS59S59WtbSKuU2aHLCaUbzsSlUzgZ6X3TLQvmE4B7lWIr/j3k14SozR4bHSK
heGj/8AQKvvzT/GhCbz/4LJEunsY5Ly53Sr53FdE2SSRbjERxjGWc/
wBxabrPh281F8tFtYaGaPtkNssYcnBMTszK/4/H3WBC8Z4J49eT/wCF4aP/ANAq+/NP8aP+F4aP/
0Mbz/4LJEunsY5Ly53Sr53FdE2SSRbjERxjGWc/
0Mbz/4LJfdgCxX8jDh7sIJfnjz2bX6zLFUpZJ6HlsNmaCXn5vMv8XoT4azQBCf+mN0UjWgf5UBJ+qsxYdQX8Nkb+5E6jOhzD1u9G+mmGJaXPTmKDFmWQ8jpB7MECmL7UavnfQyBNCe
0lMNY8CeL7pAABD0tUBKWW+BSqL2LhoHAW7CXw5F7t1+Ib5FaR8nyVjNpXKQqO2FZ+MjK6jQiPcrfcxRvz/Y4Zs0bdnsZ/NwMe2j3LXdgbEv2lbjYg4bLXbVHqcGFGZyvIsc+e8EoGiJuFCa4MBNRXQr9aOzN8JbQ/YLq5AA2wvnqzVMTR+6GLIsqOSxiAWM6VRg3mBW
3m/BvBDVhcZOOAHDnBAVtLAhTYqJuJC7Xt7+T2Di2ikvCNiAFPSQy8wY0TaafZBrOHD1T2Wwk3lJzpe1V07aqC9eHB5jSPWmKtfpRFsAryq5oeHWk+YxvH97fq2W8p9tqEbPFP3RxAUZy8yKA9wq6aV1yzFAmyhs0Xd1Ng+9htJ1WGpD+QjdC95DGJCfB0BdiTgCoKzFYBKMeHU4
Kw1nCxL2Tc2hmmBcmrHt1f0Lwh1A1Xsz2D41oq7qtbE2Z13vv1tmK5fKwz5e6Vj0pzPCkBq5EKLgDM8Nm55akDCJ4kPnrtJ2NfJvWwVjRRxzpcTgF4QpzT6ZpoHbTp11IMLqB91JCVYwJm3HSvOTH9kfLkAwi4HVUA2j1L2QxRo3YMgQmd1QTRy2OhEBTKPJhjS5WHHjKJkTzrh6NB
4EvfjqFhuRmZVDpL9A1O0wmt1VXiqqQ4GqDAV2yiT0R0dOsKj3NLUYnH0V5TxRyq/XWZXOyIUZ19ckvo+LBsD5zQKygtjX1XBnbyCbTk8t31OEBgVmsYoTZ+d7ZSDqn/w6wQjRrsqLnTM+cV9Yi/NrtSLBX5H9JPx3AZC9OvCxDS35lH0Tb2QqgVr6ACKwpl/z+nLSVX+JI9Z

XVtFDMGlZXjKFzlRtIYHf3K9K4j/heGj/8AQKvvzT/Gj/heGj/9Aq+/
NP8AGjmj3A6ZfDN/
D4OsbCCa2XVrCQTW8rFjGH3E88ZwVYg8d6tv4deLTNDsrZ48afcxzSs5IL4Dbj05Yls/
ia47/heGj/8AQKvvzT/Gj/heGj/9Aq+/NP8AGjmh3A7FLfWNM1O/
NnaW11a3swnVnuDG0TbFVgw2nI+XII55PFUdP8L6hb6csE88Dzf2z/
aDuCQGUtuOBjg9eP1rnP8AheGj/wDQKvvzT/Gj/heGj/8AQKvvzT/Gjmh3A9Rory7/
AIXho/8A0Cr780/xo/4Xho//AECr780/xqvaR7geo15/B/yctoH/AF5yf+iZqzv+F4aP/
wBAq+/
NP8atfDaLVPHPxZTxp9hktdI+pgD0znUmmrIR9B0UUVgAUUUUAF
FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABR
RRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRWXqPiTQtImEOp61p1lKRk
Jc3SRsR9GIoA1KK5/8A4Tvwh/0Neh/+DGH/AOKo/wCE78If9DXof/gxh/8AiqAOgorn/
wDhO/CH/Q16H/4MYf8A4qj/AITvwh/0Neh/+DGH/wCKoA8w/aI/5lP/AK+5f/ade318/
fHfxDomsf8ACM/2ZrGn3vk3MjS/ZrlJNgOzBbaTjoevpXsP/Cd+EP8Aoa9D/wDBjD/
8VQB0FFc//wAJ34Q/6GvQ/wDwYw//ABVH/Cd+EP8Aoa9D/wDBjD/8VQB0FFc//wAJ34Q/
6GvQ/wDwYw//ABVKPHXhAnA8VaGSf+ohF/
8AFUAb9FRwzw3MKTQSpLE4yrxsGVh7EdakoAKKKKACiiigAooooAKKKKACiiigAooooAKK
KKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAa0UbHLIp
PqRXjvxH+Itv4a+JvhrThsFnaN52okAEbZQUAIHPyqS+O+O8Vr2Svmb4ieB4R8XNFtNQ1C4u
pNfuQ91KuF8tWl2qkec4CpgnDOen4UAfSohhIBEcZB6EYyV+eSf98iqOgafcaToVnp11
d/bJLWMQicptLhOFeoYjYIR/uIwT/pPC5ZjA8kEqsmQD3xKMLYivYe9ZcTqYuzGVaSfSFQCw
hmQqrgzbbqsNBAKapBSAxCB+zVWjQuYzYv+eSf98ijyYv+eSf98in0UAM8mL/nkn/fIo8mL/
nkn/fIo8mL/nkn/fIo8mL/nkn/fIp9FADPJi/nkn/fIo8mL/nkn/fIo9FADPJi/55J/55J/
wCeSf98ijyYv+eSf98in0UAM8mL/nkn/fIo8mL/nkn/fIp9FADPJi/nkn/fIo8mL/nkn/fIp9FADPJi/nkn/fIo8mL/
AMYde0rdE+nXK7NOOBjMIOdvqHxPkxuAyMkAzi9Ti7kPBtgfQ/sMRHgp+HUBwQT+Z/
wCeSf98ijyYv+eSf98in0UAM8mL/nkn/fIo8mL/nkn/fIp9FADPJi/nkn/fIo8mL/nkn/fIp9FADPJi/55J/
3yKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiii
gDivit4pufCPgC91CxcJeystvbuRnYzH7wHqFDEe4FcB4N+Cmi6r4fs9Y8VX2o6hfahGtyQLhlCK
4ERuBlqVFmR7ABd1fnVuiRJRj6Adl6PBP+xfFu4zMKxbxY+c+qbSAUQbUbcLJpbqiJ4mvr2u
XpAcjiKjJhb05vRKTBDvGmnfxAcd4wvDOp1gt9yTbsFKwXdYqdyldB2ZRUvGlMrkDdIFbhv+TkX/
wCixNNiZyilYqKucf8A8KO8C/8AQOuv/AqT/Gj/AIUd4F//AECr780/xqvaR7geo15/B/yctoH/
AQKvvzT/Gj/heGj/wDQKvvzT/Gj/heGj/8AQKvvzT/Gjmh3A9Rory7/
wBAq+/NP8atfDaLVPHPxZTxp9hktdI+pgD0znUmmrIR9B0UUVgAUUUUAF
FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABR
RRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRWXqPiTQtImEOp61p1lKRk
Jc3SRsR9GIoA1KK5/8A4Tvwh/0Neh/+DGH/AOKo/wCE78If9DXof/gxh/8AiqAOgorn/

UX/xyuv/ALF0r/oGWX/fhf8ACj+xdK/6Bll/34X/AAo5w9r5HIf8NH+Ev+gZrf8A36i/
+OUf8NH+Ev8AoGa3/wB+ov8A45XX/wBi6V/0DLL/AL8L/hR/Yulf9Ayy/wC/C/
4Uc4e18jkP+Gj/AAl/0DNb/wC/UX/xyj/ho/wl/wBAzW/+/UX/AMcrr/7F0r/oGWX/
AH4X/Cj+xdK/6Bll/wB+F/
wo5w9r5GBo3x88G6tqMdnL9u07zDhZryNFjz6FlZsfU8epFeo9a8Q+MGgaSvw+u72PT7eK
5tZImikijCEZdVIyByMMePpXpfw+nkuPh34dlmcvI2nw5YnJOEA5qk7lxlzK50lFFFMoK8
T8Ff8AJy3i/wD68n/9Dgr2yvE/BX/Jy3i//ryf/
wBDgoA9sooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiii
gAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoooo
A8o/aG/5JtF/wBhCL/0F67bwx/yKWjf9eMH/osVxP7Q3/JNov8AsIRf+gvXbeGP+RS0b/
rxg/8ARYrjxeyLgZHxBBi0Sxv2Qvb2GqWt1cDGcRLINzf8Bzu/
CurV1dA6sChGQwPBHrSSIksbRyKrowKsrDIIPUEVzn/CDaQIvs0cuoxWBBU2MV/
KsGP7u0Nwv+yMD2rkumrMsx9S1S28VXPh+/
qjOpIHfYPWtLxTBDa614bv7aNF1E6glqrIPmeBlbzEPqoA3exUGtq70HS73SE0qazj+wxh
RHEnyCPb90VIwMEYIqtp/hfT7C/W/W/
L3d5eRqyRT3ty8zRKeoXccLnHJHJ7mq5kkKifD/
SrFvDuovJaxSNeajercb0DeYPtJ9pV5omVYgQeu1DIR/
u57V6Np+m2mlwTKxtrKLyojLJKV3Fvndi7HJ6sxP41Rk8LaPc6HxpLVmo/
5HB3KeTyD3x0o9orsGX4ofw/
pNxc6frfTT2142l3CSNaDEkkCnpjcpM7SSMFjWB4kh1Gy8zRrUmd8Sf8OqMcPFvz/
beBbHUSJLJILuSKOQ5V5omVYgQeu1DIR/
u57V6Np+m2mlwTKxtrKLyojLJKV3Fvndi7HJ6sxP41Rk8LaPLpFvpZtCttbP5lvsmdZYp
yAYAQKrepDN1PWuutvCOkwxJQJ9nnjLm5BSRP4jk5Jcd3lCgIDHHmzmd9YfT0Nc9BsnrZ
2EDEUd1eyuIR22fN8pA4DfeHrTjOKsFirHY2f8Awte5k+ywbxXf50zH9zi/BmCQqxDkGR
l0HT5tStdRZJxeW0YiSVLmRSyA52vhpzzhs1pVnJ3sM8b3sM8b8Xf8nJ+EP+vNP/
AEKevcK8P8P8Xf8nJ+EP8ArzT/ANCnr3CvTo/
w0ZS3CiiitRBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUA
FFFFABRRRQAUUUUAeAeBf8Akt3jj/rrP/6Or1uvJPAv/JbvHH/
XWf8A9HV63WctzCp8Xp8Q53ZS+FpNQvYfFosxrbXcvy6qBgR7yI/
JL8Bdm3G3vnPNeiVxsPijTI9OFl4zNra6hEzra6ezq6b1MnEzv2aw+EdH/
vp5byS40pZBJZkyyIrAqRRu3iQKflySBnkZIGa5WLTpGXAzpA0Kk1VcwjxpmLGnyQLL/
KP9oClTxLeQT239raFcWXFtcyImZZ1nOT9/
sSC9tLy1s0jKCZo2zMYkbGAQVxwAWXj1rX1zXtOUaWdF0WY3l5dSRo4SNv8AR1hmwzz/
FL9p+HOkwURxeRxLKFuTqtPhuImhZfqshBZ5VMO37tFJ9yam8fquyZ/
K5PscbvoRXMeEbp5PE+q6q9fbyZ/
Eqos8yDqGnpfqshOIofMkVOO37tFJ9yam8fwyx/
K5PscbvoRXMeEbp5PE+q65dzyLDqGnpfqshOIofMkVOO37tFJ9yam8fwyx/
Eqos8yD5YGTAmYnsWhwB7rV/
URd2PiDWJ9NtPMmh0FBbRhLaRhPlaRXlKoPfP04oESSf8Jhcx4kcx2kO3Gg3MOkTMmLlpkLqrkBXaPOO3GAB6cUDo+ITDKYvKc
vIPUkA9K1B4gtY9Sv7G8BtJLSL7RuBtJLSL7RulICyw45kU+gOQfT8RXB69eaZq/
hi4Sx1PU9Z1FVSSSJWdREFYi8SAKuAD8pGc4xzWtr9hL46u1XTxHHa6ZiWO5nhOLmcgMI
uesWMb/
UkD+E0WCy+ymZfLbxmfLYzZaNJhhymfVYjtkc464NXaz9NS5a2ktZW5J/b2VK9vJ
96KQcMp9fYRrQ/+jUrtfhx+H/
wBBFcV8YP8AkA28e/XuhYXsWBQ+U2CdR8s6t+Ju/g/zP9H/
wBBFcV8YP8AkmGQq/wC9D/6NSu1+HH/JNvDn/
YPh/wDQRQRWkNjansdRRRRVGgV4n4K/
5OW8X/APXk/wD6HBXtleJ+Cv8Ak5bxf/15P/15P/15P/
6HBQB7ZRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFA
BRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB
y/xC8KHxn4LvtHjkWK4cLJbuw4EinIB9AeRntnPNeR6N8UPEngTS4PD/

iXwleSTWS+RFMrFN6LwOdpDYGBuU4IxX0JRUThGatIadjwv/hoFf8AoUL/AP7/AH/2FH/
DQK/9Chf/APf7/wCwr3Sis/q9PsPmZ4X/AMNAr/0KF/8A9/v/ALCj/hoFf+hQv/
8Av9/9hXulFH1en2DmZ4X/AMNAr/0KN/8A9/v/ALCj/hoFf+hQv/8Av9/9hWl8a/
iHceFdR0Cw02Q/ao7hb+4QMVDxKSFjb1DHdn/
dFepf2zZnw+NbSTfZNbC6VwOWQruGPcij6vT7BzM8b/4aBX/oUL//AL/f/YUf8NAr/
wBChf8A/f7/AOwruvhv4lk1qHULa6I8+OZp15/hkYkgfRs/
mK19c1n7Fq9nEp+SI75cehyMflk/iKPq9PsHMzy7/hoFf+hQv/8Av9/9hR/w0ATwng+/
Zz0HndT/
AN8V7oCCAQcg0UfV6fYOZniPgXQPEvjP4jr498S2D6jr498S2D6Za2keyytXUqzcEAYbnaNzNuOMkjH
Gce3UUVskkrIkKKKKYBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRR
QAUUUUAFFFFABRRRQAUUUUAfOelarZ+EPjr4rTXJltEvHkaOWQ4T53Ei5PbKnrXpH/
Cd+E/+hj0v/wACk/xrovFHgXw54xWL+29NS4khGI5VdkdR6blIJHseK5n/AIUT4B/
6Blx/4Fyf41LjciUE3ck/4Tvwn/0Mel/+BSf40f8ACd+E/wDoY9L/APApP8aj/
wCFE+Af+gZcf+Bcn+NH/CifAP8A0DLj/wAC5P8AGlyE+yRJ/wAJ34T/AOhj0v8A8Ck/
xo/4Tvwn/wBDHpf/AIFJ/jUf/CifAP8A0DLj/wAC5P8AGj/hRPgH/oGXH/gXJ/
jRyB7JEn/Cd+E/+hj0v/wKT/Gj/hO/Cf8A0Mel/wDgUn+NR/8ACifAP/QMuP8AwLk/
xrjPE/gb4beHfGnhvw/Jp8u/VJGExN5JmCSYZ6GF/9k0cgeyR2//AAnfhP8A6GPS/
wDwKT/Gj/hO/Cf/AEMel/8AgUn+NR/8KJ8A/wDQMuP/AAAuT/Gj/hRPgH/oGXH/gXJ/
wAKJ8A/9Ay4/wDAuT/Gj/hRPgH/AKBlx/4Fyf40cgeyRJ/wnfhP/oY9L/
8AAApP8aP8AhO/Cf/Qx6X/4FJ/jUf8AwonwD/0DLj/wLk/xo/4UT4B/6Blx/
wCBcn+NHIHskScX8WPGvh2+8C3Wm6ZqlrfXd+yRxx20qyEYdRlgCBSSR0Hy4/GvW/
AdpPYeANANtbmNo547DBU7BkEeorG0j4PeB9F1GK/tUcVT67ScE/
UcV3VUlYuYMeVWCiiimUFeJ+Cv8Ak5bxf/15P/6HBXtleJ+Cv+TlvF//AF5P/
wChwUAe2UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRR
QAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUVT1HUIdOt/
NmbGeAB1J9q51/E8zklLVyVyY5/8ArUAddRXH/wDCSXH/AD6v/wB9f/Wo/wCEkuP+fV/
+v8A61AHYUUVx/wDwkl//ACKx//FD//
+v8A61AHYVUVx//wDCSXH/AD6v/wB9f/Wo//ifcV/8Avr/61Hyf/wDCSXH/AD6v/wB9f/Wo/
wD31/8AWo/4SS4/59X/AO+v/rUAeQ/8Av9/9hXrUAdhRXH/8JJcf8+r/APfX/wBrUf8ACSXH/
8A31/9aj/hJLj/n1f/vr/wCtQB2FFcf/AMJJcf8APq//AH1/9aj/AISS4/59X/76/
wD6/wBpS/6lH8/lKH0/0G1EzKN3SkAwDgCgwYmGfF7nNSt10EbGeKFX0/4fnngCvd/
wDs/4tiTiP4Td8aGZnHYjPxi0FACp1g85Bz4wHjtrGygQvlGgSb71oX8Xbi5iUf8ynnm/
wAJ7tc/L2v1DcVzi+hJvqmEcgnP0hvyZhsP5vkDuMSVN5M3NcK6GmAYg0BsGJS9ZxJ5dQ/
61r/8Ab1Q/6FqNPygwmoJYtMmiU7u9i6WOM/N38MZMzmgWZbzqL3AtFjDCWKUwl1Fwg/
w8B/KrYSXNDJpu1qQ9XWyzuP37+nGQ9MXUQrV+Ngwz5/QP8wJsBWRd+vkbNkvqgIU2SAc
KPgGyOqrgN+bmyBJs9WpdMh5SRbmxQVzOJk8uXvhDrkuuvwvp3tpzvxkmMpd1g1Si
iS96uXkfpg/v7n5b9fMdKQDWiQcU8+1VkQR7DXN0tXtmt5sW5ek6cP+fGhDYm9Xe9SK5NswI
zWr9MBMq7wgN3cY8UEyAR6UMB+OHzh6TpYjWLs3ue/yUoOwsYqD3wyBrVX3n/SDkgvUGu
bUpEjG3XPM/nWx6mfVvMo8Qvqx/5y+cxiLNbF7tEr9tpvyZkLrpnFyE7b0IIA2/h3MwMh
TcaeR3xCDtc2KEwQK5SAHDrhs65PivRLA6uMufmqEdy+4kFFDFxIZyHk2TZJ1xI/W46
cEDXzuTMsrt5y6yECdb0uaT8tWQ6WudnSbA0o1nGf/OLIFfQs/Tgr6Mv5Or+0KJ5/9j
1VXjRtLQ5MYLdlwAOxUYqHnXhiwDRX3mBdOfBhR9kdhwyPfhxmg+hrqS/l8s63Epg2S9
HtklsX76nzMgRn7JXCF6aZgbS+Y9uDywMr2qCuHqbWE6vu3L2hZSNbv0Y9O5C6LNhO+P
+lZqzvKnDk2s/DLsxM2GgZGd5a7bGrQDk7BGR5V2q54vrhZWaL1e3y6bCL5XmxLf8IiG
SYZ6GF/9k0cgeyR2//AAnfhP8A6GPS/wDwKT/Gj/hO/Cf/AEMel/8AgUn+NR/8KJ8A/wDQ
rQyJjDbYWX2Ro9ghwXsK5yd/r1sw8LL8exgMSD6zF6e9bNdnd6FaZGeYhvHWHSiVkEiD
8A31/9agDsnC+1y9c0j5E5DAzqG3+ps9CQrb3hCnqnRIsXuA0m+dfXDU7BP2wJ39U3u
x7ghYpfw8/mgxtd6dEvgsHBWKW9IKhEUlrGhGuO+fRhC8ShjsPJpCBu5JwNjkOtjNtB2w2
Rnzy0mktzHmQ59VrhF8V9QwBVLbvyIwfSjG2A55QwLCVpVn+yZKsgmqIiDxyohA

AP6GKL/wHm/+Io/4XJ8P/wDoYov/AAHm/wDiKAO6orhf+FyfD/8A6GKL/wAB5v8A4ij/
AIXJ8P8A/oYov/Aeb/4igDuqK4X/AIXJ8P8A/oYov/Aeb/4it/w/
4w8PeKVc6Jq1veNGMuiEh1HqVOCB74oA26KKKACvE/BX/Jy3i/8A68n/APQ4K9srxPwV/
wAnLeL/APryf/
0OCgD2yiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKA
CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKr3F5DaqWlcKB6mn3M8dtbSTSMF
SNSzH0AGTXi8NpqHxAvrjULy8eCxSQrFGOcewGcdMZNNK4HY61rVneeIbSBpUaEKDt3d8n
P8hWwNWtFAAkQAdACK86b4b2UeoxSjUZyVxxsHvWj/whVr/z+y/98irUWB2v9r2v2v/
PVfzo/te1/56r+dcV/whVr/AM/sv/fIo/4Qq1/5/Zf++RT5ZAdr/a9r/wA9V/Oj+17X/
nqv51xX/CFWv/P7L/3yKP8AhCrX/n9l/wC+RRyyA7X+17X/AJ6r+dH9r2v2v/AD1X864r/
hCrX/n9l/75FH/CFWv/AD+y/wDfIo5ZAdr/AGva/wDPVfzo/te1/wCeq/nXFf8ACFWv/
P7L/wB8ij/hCrX/AJ/Zf++RRyyA7X+17X/nqv50f2va/wDPVfzriv8AhCrX/n9l/
wC+RR/whVr/AM/sv/fIo5ZAdr/a9r/z1X86P7Xtf+eq/nXFf8IVa/8AP7L/AN8ij/
hCrX/n9l/75FHLIDmfjhfQ3J8MeW4028YnB/3K9b/te1/56r+deCfFbw/
DpZ0ER3DyedcspyBx93/GvRv+EKtf+f2X/vkUcsgO15ZAdr/a9r/z1X8qz1d0+ax/
89V/OuK/4QqiR/z1X/Oj+17X/nqv51xX/CFWv/P7L/3yKP8AhCrX/n9l/75FH/AAhNr/
z+S/8AfIpclWticCRfzq1FcRzD5WBrz5vBEO07L2QN2JQN2JQEVJ4bvbyy1eXSrxy7R8oS8f4
YINJprcDrPEEEb6cZSo3xSo3xkbT9TjFa+iMX06FaWEm7OSo50J3/N4Iaaw+eSkuyrL1s50aU+6/zFaWhf8gyD/
cH8qiW4GpRRRSAKKKKACiiigAooooAK+f8A443Nxq3xI8K7FZp3TITCkHUcaoNChIUYy0khJ+p3U7/AIVf4L/6Af/AF/6AFv/AN9v/
wDFV11cR4p8SalovjfwtaRm1AZgoLT8WhlCbkRsPMVZsPCKW8AtMQTlTIke/OMjaghD7xEcVWeiMjkxNY8X5KgKSVfvPGUTAlxwe8UNd0owirtFn/hV/gv/
oAW/wD32/8A8VR/wq/wX/8AAt/8A/vt/8A4qqauth3jWdUHbmVaEDIGsjhpxE4YTMAlmMQP8Wq/XP8AIgCGaGnfR3ff9Ar4QuFIY11XiPn/K+EgrwXUn0zZSV/wt/AtPyEPyXEp2XzvLedLJTcPpNtjS8G3AbvCBxGfAP0yHijVPaW8OibsfXBgOqAewPwwhLt9KEYEdy3/Ar1TRCNxl8TfiGNcsQGCPPEZHvl56fxGrycDOFrKQBawJkMhn/fFsaJrbpmtAm0Q5QbKfWsT9U8wXrG1WkKqnR/m+5adatTGTSP8VAEXvyBuOi3XW//8J8wgyjWEhhVWbfRaOT6+ttbNkp6oMyBQYEIPBV2BzOUqOosbOeR1K+PgLxOytbO7jIHVzoqIvrM90NHPjRrfRIhDEiN2y0BXZf0dS0XNh9RBdlmujnHjjfmhSlHHCaMvVoVMJ2V3ILEdGtQc3I6YdPf3q36YNn+VCAMl6L8tEH7MgY3dtbEotl32jE8PxuhWnE6cgvmqn6kvAJQt1B7nZ6FwKKkfDbB8wXB0fZjWQKWnxiRa3XBrxdijH7JaS3XlMbTI5i85GDMI/Av8fVjGNPZ2l0TIQBcABkDbzKI2unJnlfEDMYnTNkUbG6Hz83MGStBW7qL4ZjKaAfzG4TPxl/BtAEnPUT6BD2ukDkT6bOYnGM0Dah5J5dT9MS+LQaqWwqlCd+J4jMyaOdJJRSbpK8WZyOJnzUM25uNTNN1sy4xfVDS7QDaQDa0uqWWwGufw1vfkxcj5zDdoT6iOWiyw5zMQJS9ETFhc3e2A2IW6ETdsfDDBBCxF+vTdc6OpJr+Qme4WKFRWASK67RWLYkKcz+qfkVCkMR9RfZu9ZlStkRpiiVfRszCxmrkWs4DhWZp3OUa5Qvw14oOHQd8Vhmq4uWwFg1HxM3PFybx5b6ub2a2LufyjbR46ANKOQdOqgnC2SnGwGqcFn3uDT5mqlyFvWSoGvSQ7Kpt2f+MqAMq9WHaE5RdfHJ8l8eFcM/b5s84yNkpsGqy42yy28ckp/GHqvqD16Kud1GvHNU58/Hb8JMyPdsgbSQGwgs4rfHObfASoNcw5Jgs5wK34OJaqaYhiTJ2dI8uafqZzu8fkzDi4m7ewh1RZAA8bi6h3s0p5VtNPtmBXZJJmPyeH3d2eGk+g9xfV7XCBaBzJmZwhrTi71VCcJ/T9WTUX8kBmgt14GHmtmqkE8tUxMa1JeVmmoRxuEfKsg/Q1TSR3tFSXUS/wAsMbFXtf6oq0xCgGU6OS2bbfKF4xmCeEOwDJaJ96mZY3nk+sMe3QJG/H3V8zQ3rZL4UTrngdCvKFeiTkC6dDMSNHIjzJrtg7q9o0sJ39dOyBNtUwVNu4DtyH4hZFawxwzFgx8XFrLtNI8WbhFZOT/ooinJWZjN4SOwbUkvn1sTy5KGAhtuUAtTxjg8hbHTH7Z6h3oI3lWz7Yvr6xqlWbkqf8I9zCxsJ/sZB4R4Yq3SQsjXYxJSkbU6GrGnN7KFdhDWJaSkG4hGBE/1A6Gw3v2G81l/1GqmP2TeOuMUZ/qIHHfzJFL8GLCHzf1PxcEsTD+WVcY0ghA4wFmC2aJ+vNcNS7PmqSK3nOnEYFyyNYq4zqnhmVcJRVa5Ov1kdbxjqkkt8Oit3qjqNq30sVyNXcUw4TUpmfHrnbNd0yg1DIW6Q9ZMNc3j3yXMMvHN+zSBH+Q0kXgHO/JrxuygCqO0gP8EY3pe4hcdC/Q7gxU1FLdQkxWmRVxNjgDuUIA0hgmcmIGHC6fHtlhbq6jNd9M1H6KuT0Nl31lHH5xbECP8g4HIqRq4ZKb5ZG3gXBl2VaW0Yh9QPmWgRhGIh8mOl4ScSn0vDgA5o0Q8SPCLjSB5p/5K9hrCCeBQhH1xQAVCYtI+aR6fJK69bL6n5TjcnM3xEL8m4+K8e7IH+Mm0Y3FoTv6C3IdfcbJIFp9h5e4xJH/65hCQsW1lQx71Mab0U8v/X28Mdxx6KD3D1Da8Hq8XBgkQ8tH4HgyhnDYGtcKeePLqcgqT4iRl4zzdSI0+wHbYd7y5yCeq9gzhdSvmmSr3ZYfwMm3ZbPnYfcQqsUJ1ZGgBVn5dDiL7aNn2NRVXJMlXr1AoAqgF+NZ8oNhcB54H/FJkBNhzV/EGRNy7xDo5Y2qM9XFKVoMPNG0/y7Mn0JMbz/AQXeM/4iE+zjl9cI3YOoBMqKK+8Ul0RUvgjjVy2QbAYRgFT8yMySO5qsrGAAXNVJZ9gqX0Tqa3/AB9M/WnGQyiDAUvWIbDJDlpP7IgA7TBKG3JsF/4pk8XeKR3VuiHdLdDoCIcxN/k4CGG6DsfvRZ+Y4tbA4/9P1kb3uc7TRc9l3eWszEP8jKaIk+Lk8HrO6H0/fyzn7HvV/4HpAKVaR3UNzoN8FOI/4AMzJmOPgR6XSrn/FC+ev7mfaQcfEpjMXFI4ubsYZeNqHNF0tS1tvw4bBNwAD8b2mcPXcqRnDeK+CCZZBJqgZnlP3v79LrBVNrqIXrhkzBCyA+rumBtrnfZpM8O6pMEtMuK/yYoGd5g5jsQmPeoD1/OpKSxXH7W2m+m3Fg8Nt/5ZOSrYz+IFi6Aly/yN9okAfyEeGzMt1l3OOT4wvobfpYOqEYcXXsgbY6s3BBRhgtQEa2KqXc3zBdXNZBUIjkM6ZHiGFcKgRIVFsBiQUifxk7BQw+mGYD8F+3TkU2J/Nky8rQThRnpbxKpFAaifeXxfNQq6CyDpNDQLXPpNqWPD6lsSQfDsvVDiCyl9A96F2hJhlJ9NLPR2LsJYP6ojkXM0eYQ5DHBhk5u8dTr8oVY7qZgw4VvGlNMYWKT6iIVb+6D9Vj3hJTLSnY4UXLpf0mQNYhLVQ6SZSrM0yAVtH4mJkcoQRpcx4Bbn2J9NBeuUmSzoTVjZs0cWqGCc2u9ngR5KHOkLKxbiYgzxaDTo1FUoQCcOD3yv46u9BlHt4UopnXTGFZGFuAvm6n4UYS5sddkp8HeqtJyA0YlPu/zdDZv6ZOG0e4u3wOdcaGVWZbzfHD5iXCQGtbojqYyfaXfDgg6QPcqrd41mxeegDzHmIyyd0WnJyeGpqjjNjmBi3JmbxWbyn7JrhbHjJ7Bh8f6mHMJ8Q7TrEyrr5N2KC9MG0bQMDjdvSUzwm4Xa6kFs3NrxV9OqDdytRWkpFyQ2izHsGcZ82I7mtaXFzBPbT5y0cgD+4ptjuKcnkFk2gz0JU6jQBuaEcbiP2nJ4TrcWeGt8wjaUCJTpzm0Qh3jDU1ubHvQsP8Kfz5IvqpPHCFQCYWdpt6iDJuDsPATfqDKz9aQSY19m6VmaAjC8oXJgb9fYlEkfoePbVvCcyaEmwE7LNmxDDmchDIatcOwjNH8xh7vItrX1QVEXHKExJnnPgK9LMlIVhnBvPfh/aEjwWZyIWw+WC4zcr9TnmXnJx0qVsbijzT+eD3/0YXc6Ej0yqAhtlEqivkrtvIyUStnfGkWv0AytPCiimUAh+Nk/4qF8Gk5szlnUYXqGWZTKOTl4tYytVE05rPrEgBaPm3k+cG6lJAZA0CI4h8PJQS+oESAYHXYz+y4tI7x9XNsQIUz4/kVmrFbnBX4q6H/bKqPvVg9DLwz5EWMkCqVSB8zhuBUojwbV9XgN2cLc8bbfbgw1qQDgjcP7JpDhUI+0fO79VKIlwgtMbLGTV8Blb6hRhEF0TuP5EhBEhwKW2jg4G/cB5xDNRwKIN6LXeyaHCrOhXhR71z6uH4tlhjYZTL7hd8RH1qn7IZKr18nhOC7TMX1R1HchTC8F+9OXW2zSvgXoQ2PEgbs2Al4ivKEZswtyK3g0IELBwSfQdKuTDCXSa9TWJbLOI/3kmT8i1pyTd7Lg7bbsjkBQAa3E3e9XFgLjpK2AFRPQSOWlQDyQdjFwcVFBbvD7pEqI/0+xGDZsqmEzUh4x3nB4VrPE3ARlwTxoQ/q0HRwUkYWnA5t3Ri6hEo/5p3Uvt/vIRzPwh7F0BBr2NfSYzn35rSaqYJm+mBIbLaOIIpr9UKNjVmhC9fiTrgmZRDN0D2kC7iKdfQfaigvUtUxJSwtGdmYuHBiX8ORa3tN8rHVMy/PyHQVYC4dGEhDS4C4r/jAe8TPaKW0UaEeqLAJzH8f7x/oHoEoe1jdgGOB+0+PU9sJ78yvhF5Izeqw4h6QygrbgKC/eMytVIgUNKLbm7RBzAVkyIzbcgqjJJOnbv9M8gN5TyH+XBPNaqYzZGaZ5QNNmPbAmWBIlbX3Ic0MRs0z9e0zgB61r+IMudRMnv+H6gtB6JAF/MpX+TjOa4/aJUyzhaxfvpi5e1rzL9dMGQ+m+4rpCe2mUXxV4FqqMaLZDhg1HaPSZRBkbA1ONxXo9zP/Cxl52ba3WFMuGj3FtJFyR4Jf5/tO4VBZDxnzbGs4qklDsU1fXdTNbjzFNq3MRlInB/MuyXw5tRvvMwAXsKtkjeRw/j5+g2nbmm/+HaaQpkbnpOFMdX4q9i2yP04XaYSQfjxH6Y5Gm4z1JMfqrUEabaYP+L4jnVP5bNmKUJvTx7hR+Lj6W0mOAKtebIEYYSVy7kL7SXmKvbnqlOUGNdz1qTQlmtUDbDt3uXEGsXOZbfa4g3HqeqWDgo+2HiRLXHr8Gtk6LW0q54QXPiW3XMeMCDhtIcUsL8XRKXxdTgM5SXKy+B6jtvT1iF5gKRS1+AMlqS1oGPnPCU5Cnt0UDzNp9aIRhTXDumOwnNTRHfAUKIbkmHgdPugJJuOgmD0tC+E/p/gQOcZdcG6kYJxKqzWgHKCj3KVVdBh9R9HVlBvKpB3GvrnGbF9qL1UQ1oLgJDCK0ZMbBIsZ/qAxAYtAqjlLOT2mWrMOmXd/eRZvMx13PnVUYmYMTsRPCPFjNghmnNq7Pcb5iYKXxtdVUQXNTKkVyj5u33L6OkRhyrqvaGaJE/yrMc9UdVObLXHgYb/7eS/LlhqJpx0+ZWEDXb1TwfsG3m2v1t7PVOZNlwMV9cEp6+X1jgGvCwlLtlXnGhBdS8mqsOqUv3eEpxNehpBMy8p3VUCXPStj0gMBOMpjMGvVJOMDcU5JnR9qQQGaWAMgKlWB1ncqyjRNoKqbtuMF8j+pxgd3m+uiNd4lPO6s5JmUV6MKCyOLqRn27yZV3t9+g6wJWqW8V1D1eU+DXsflkmF2pWyNOsH9Akc8/qy+MI0KlQX8C8JKIp8UeDUXYX/jfEw61iGGK+3SGITqSbHjrugRGflxw5jOOhBK3FN9CzdjjoCoj8xo4T03TCSmL3pbnuNLmTAH9+Np+GInCSEd+wWSzZWjIQP9QZVwL6M61fsfGaaSY6x7jXWyCZ5XbhGWA56VDVtDWbnn8XrZBtz9yA0wHsNYoOVRaxgpnPCUcCU0P4iP/2Gd8bU/5N0W4mj/jkY0A/YvfzBWqn3XkWJfKJXhOnAAbtmdGrh3aNLNqOWDjZ2GpjPaWKlNTbwQbhT3SQJC9HftEStlQ1BggxOHSNnnqBxCiNKOKfcsz8QlTTlGzd7PLEUzyQ+2pw3pYLJiDwMyDdGDh/pxNGfrZazBuMcwA0pKZfmLqIFqT0fRuLJoftDQXfwm2mpYOzKfZT6s3XM/QLrBaOV7E7+Qo2bTUEdhUX6Kj8fE3PyEtdzg+MjIhSN9fQ1BJLBLNmyiaNIh9Krpkt8LXnw/0xUIi0fnGFKPXH31ZpADdJxUEcDvgRYoAcf1C+6MW42d1gBmSrjIjPpbtGTrWlcG0lNXS6GqOFhJTTz7hhXIYF6zwGwLiwbUy/1mfvGyr2JnsTJJYG46jV7kPcy9ARBRnz2a/LmW9a+FRxP7ew4a7Fv6NKvaMiBq4iHpRpHZJAdCGPjWg0SIs8D0fZn84cvuVKoqabvo4nxQQ6/XR8C5YnGkKbmUKxRJjMAq1N2gs4HX1hV9lgWsPYVL7DXu1mzoC4JOpjbqyBpCOtRJF7X5JWkdWDGE4KTwK0BUEqE7DPmOVKxV+x+Q1bAqL9UdSt5J1eIDHGTPDb/DeHtUXoOaNoLqkmIrIZxcwRo1WOE+o9H96+qCb7j0XN1eCmdWhjvoZe5ITv5PW+KrcZwoRnFS+EGeuc85DwfYMrnK/GzaY4HnnLHAn/tZsO1eHQ1NN+Mxm1jeH+DWBJKqsDofrIi5nyKkDmRYeuD0XAQvCI7C/BgZZ7cb3RMU2aaLjGzA1c+d8AE3nNIrOFCGVdsfhAH9v7lfKhLq24cKVQIbeZS+PZnVx0Tu2gOtI0D7FiJNzl2Mt8LRefjsz4KAuG3Prfy7SnhJLXpxnfuk6UZdvHTJsBBJAmUBDQ2I0ESSftjgM7Tt5sVYfqwZRNadQaJkC+YUkKoBsPLQwD4ocZSOxgO8ooL+FKmH/xLZ+hCX0Gwj9yGBQlx8A+bZQ0z8gEWMRx2x8hG6P5l6kMYc/bFbsE/Q6hNYJ6t8A0BoUP/W7Z5H0yfoUjR8SXMd4aqCtlcaBbiApZH/sLYS9oFOnUHeWcW25zCLRJ1p5uGe1+3IcRb4nnh1q93Qlo5p5GuXA7x/iiTqGTkE8zhLgKi+M2MNR6xz4Dxf26/P9W0ioNQzEbgaL9UCzdQjDdt8HDIZ7Rkp3qEHKbbAjz9wgIbsNDT1M0evqG3c7Lrlox3yWg0ILXsPRqWQsOV5uuX1Dfu0J4dnlRHqFRfH6hCHSXH4TJ2MBbnw73O0xwdvSmAqWgsCjsMlw0rCzDYTTkvtiSr+5JTYObaOGbtk5hTSbmJG7Uthbqmm8tBfB8OHOITn4Ke/hyx3B9Z33LghlHnoVIyGCsYvzAB5CZZaUQmhGkB0iMdtODcq4cEJXJmY2Jm3wKEwXBeMkDvDFvAQVeJjzWhuWrMFmf9vN4DB+GnNw5iFpgaN5SghqUv05VK+kfWeA20qcLk/WDi+ccWoGQu2MQf3yr3gcvpbR4ZuaRlbtR/NcOk9AVN+vWKnhcPCu9dApfFQ7IG9FyF/n6JFO2Vmy0b1YnXQ6trAGP0ZGyxeb3R3D7PYgBTxE9cC4FUaqhkoUXDNdM0mxBPJ4KUWA4DIx6J1OJgV9sv6yaWsYqVW/VEZBdK8kHzw3tOcF/u2m3nrxmtJsXqZcGYk4OmVP2Ztr0TnpqxPfM6UItOEBOqIOeWo2sl4TZk0M87ArjTDsvuZmPrtc3HuqBStDUn5khOfK8TbSk4aI92Kjlqak9FsfIOT4AdDmfNpaSmhB9QW9jIlTcHRMQW4SrPh9W4HAJIDBK4/mqXB9A7BPJRZ24I4kECjTGtsYy/kGJwNj/Zd3DzCsXE3pAM9a1DGStwoHmmaaIN0OB+w72mt2EKrc1DJeyqEeOzjQjK/2hQKc+QFz6KY5FhgwK/wU7NqoxTxiN3gILXkD7ECiEhlC8omYwIEDt6QPVmDoKKO5ZFn5xrgbS6eWC+WQ23k0RHpT38aX0pL6i7yhkzJyJ4lvf+F4F6V6DqRAyqETVOs6Z6pAGzbCvkGYvG4GVmG3A3GcW5KVpyB0oD3fCfJ+lnmUHQOTCjTuuK2jQMCqsFFoaLV3HeDIYo7/hcZmYDvKg5Q9ECJFhjHo3N1JGOfDVVu2KkQyXU0tgtjgR4RO4S1fE6A8rKUCs7pzhAmEA3RPB5lk6fi61D3K3Fm6yOCMGjXZzONI9YmiaZu2NaCzmK3Th+ymsBXj8RtgB+NOpkcpeQJpk0I8ZIXiXKNjx6Tq3KDU1hNg6L1X/WqiR8bW/pxlcIl27zHFb4V3IOcfmOAKfUvwcQjnA7+MI12Fqe4c6L5AJWV1t2t8XPQn0nF4A5KNSM+GuGdw0mdXPzh6pXoIjdNhWZABGbBHGgR4i3xjCnb1xHJeSnOyyDOE7UZHfWFu2ljl0nFvl7DCZ3Xoo1gSrJw8+kY5AzLJuDO3TA0H8Xob3bgFsKEgzOBdzN3jHzg7TbJMvTdIQs/o5E+3xsDgaAHeTCs+Tn/QkEfuu8CbI3FRGxuwP4xnK5I/iCOy4bR72nSddDf31YrIBvnCsibtYIQ8q1HjItoSLSgiyOa4D7vC3Q5wewufEhw+gEXvaYFhm1kxIBU0fp8Tb9FlR0cpOPfcs3q2w7w02rKq8g4zywWU45H6XbGZf5VfBlKXpjvTiPJeOkDTmH8TgFV3BXF9HsLgSbapa4iXOYxjJmVYzd7/9BYgpQaLM5AsiU4Uiou5lMUB0m4prxYxn4ZTONtqTpIHDA5kj8JGhRIxRBewIP5AdsM0Abyc6d9rGBfwJuMQvwmP6nNKbJ5DUnxbsDwYsC9J3KJydsrhAlrqOeAYtl0fWDFEGi49Nb9w3DKAXh13pCWIYVCDhGnlx9p1JTW6JdAyTFaUd0c1/+PP3ep1wi3C+ICWbMtLdAMkBeLKR9CNRFWGSS9+O5vFvE8wsvHbz62VNnnF38AUKi7t7lSd0+HdWeAHiThx6h0+Im2bRr6Wbf/OCRb4dXAyEiD+6v49BcbLi4q5IPxI+e0QEaqkGoH+4HM7Q1YeYnVM44aHyr+DLPQ03U6SiP6X/1wc4LbW+Rh7tQMhIIshyiJpqnsYZDX92o8PgXp+7M1ZPwhaX1ppUQAAAAAAAAAA

Y9q9ojjSGJI40VI0AVVUYAA6ACnV0KlbqB5G/j7xg0gf/hDdS4/6Zyf/ABFL/
wALA8Yf9CbqX/fuT/4ivW6Kfs33A8k/4WB4w/6E3Uv+/cn/AMRR/wALA8Yf9CbqX/fuT/
4ivW6KOR9wPJP+FgeMP+hN1L/v3J/8RR/wsDxh/wBCbqX/AH7k/
wDiK9XFzA05gWaMzKMmMNw/CpKOR9wPJP+FgeMP+hN1L/v3J/8RR/wsDxh/wBCbqX/
AH7k/wDiK9boo5H3A8k/4WB4w/6E3Uv+/cn/AMRR/wALA8Yf9CbqX/fuT/
4ivW6KOR9wPJP+FgeMP+hN1L/v3J/8RR/wsDxh/wBCbqX/AH7k/wDiK9boo5H3A8k/
4WB4w/6E3Uv+/cn/AMRR/wALA8Yf9CbqX/fuT/
4ivW6KOR9wPAvFOqeKvFBsDN4V1OH7JIZFxBI27OP9kY6V0X/CwPGH/Qm6l/37k/
8AiK9boo9n5geSf8LA8Yf9CbqX/fuT/wCIo/4WB4w/6E3Uv+/cn/
xFet0Ucj7geSf8LA8Yf9CbqX/fuT/4ij/hYHjD/oTdS/
79yf8AxFet0Ucj7geSf8LA8Yf9CbqX/fuT/wCIo/4WB4w/6E3Uv+/cn/
xFet0Ucj7geSf8LA8Yf9CbqX/fuT/4ij/hYHjD/oTdS/
79yf8AxFet0Ucj7geSf8LA8Yf9CbqX/fuT/wCIo/4WB4w/6E3Uv+/cn/
xFet0Ucj7geSf8LA8Yf9CbqX/fuT/4in2vxWvbK/
it9e0e600SdHlDdPUgqDj3Ga9YrkviXY2974D1Jp4lZoEEsTEcowI5H6j8aTg0r3A63TL9
L+2WRGDAjIIPWsQf8lAP+4P/AEAVkfCa4kuPBli8rbmCsmfZXZR+gFa4/wCSgH/cH/
oAqZO8UB2Os/8AIFk/4D/MVpaF/wAgyD/cH8qzdzZ/5Asn/AAH+YrgPB/
xE1aw+Ib+EPENqFt7t2Om3G0KQnzFAccMpAwD1z1z2zluB7LRRRUgFFFFABRRRQAUUUUAF
FFFABRRRQAUUUUAFfPfxc/5L34Q/3LL/ANKpK+hK+evjKwsvjX4S1C5/
dWiR2paZuFGy5dm59gQT9aBx3PYK47V7OLUfiJBZTjMNxoV1E4/2WkjB/
nXYI6yIrowZWGQwOQRUZtbc3S3RgiNyqGNZig3hSclQ3XEIBx7VR6DVzh9FurnXtSbw
iWTQw0mosf4rhCYos/
XDSf98mk0252aPotnFBA95Pql61tJcKGWtjJck+XniuGlTaaaG3ijjlnIaZ/
0QBpCBgZx1IAAVByMEkzzniu4itLa4ISaZ5SxBx1zniu4itLaCaaaG3ijjlnIaZ
0QBpCBgFiOuBxzVe50XTLXyvbVX9LWQ5kkLBAzve+w3Gn20lrvMgiMQ2hiSSwHY5JORzkmgnkZyl1NdW/
jnRotR1X3d74kxZW2qpmP92Y7zgBhAfGHj8gaBqVrquj/Wpvvarf5o0sQgoB37osSSmmmmmmmmmmmmmmmmm
wz2kXkKmYXttu027rw0ZHYqeP/
rVfSCGOaSZIo1llx5jhQGfHTJ74pIra3goqnnmJpPAzkhNi3P3Az5oGvzdS6u2z6bHa3LAsnaKqM1mnuiCu87y9NmGbP5
1ZY2AmfSbfyMnuJJ8/qV/MVF4Lms4vh9YeyIkFtsvRKfuSqD5u
PQ7t2c1rXemtJrNhqcBUSwkK8MoYkb4XwS0OnUMqEfj60y58MaHeXpvLzH1I6F
h0Y+5oCzucN4de2g1jwMdrumqanR8vt6onbkumRDEOf+meAPriuo1q8p+J3/JRfAH0Q1/
MOXKL8z+mT1P40CUWtDO8LSaguUmpbuKh3C2BDU4stOZfW5p0UpBpaLHsRDh0SnMpLncNRsZUdDagzWmMOJK+ytNR10+yht/
MOXKL8z+mT1P40CUWtDO8LSIrmmagzOAI9Tvd5z939+55/Co/
EN5Be6PpM1pVMksdzqVk0ZEGHIcecjEj/AICrVsR6Xp8N1qxclDE8qwqHlHoxxk/
jVSXRY31HSykUMVhpwZ4YItz7F+UDAVVL492HTHIOztY1q8p+J3/JRfAH0Q1/
o2KvVq8j+I11Bd/FTwPptF3MF7G0sanJQPYPpwpP0pMit8B9A0UUUjiCvE/BX/
Jy3i/8A/8Ak//AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8n/AK8/
wDQ4KAPbKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooo
oAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvDPjX/wAlE8C/9fI/9Gx17nXhnxr/
AOSieBf+vkf+jY6a3A7eiiiuwYUUUUAFZfiVL6Tw1qKaqKabv8AtjQMIvLOGz/
snscZx71QVl+9Qj0ieXS0El5HtdIzj94AwLLzwaHsBy8Fm8gSCTtjIG1g:BTpyay8f/
AAg+pWo0qMWlheFQJLH9nuo37EFgG3kOzCM/2fpKJ/
aWryQgt83lKVUANK552rn7fa7oepWT295Dwj.5SvicY8ZgFuAmFrnQ8ZC+ucd896y9M
sr/
wAO3el6nqMcNsLjxVlNetB2NPrtqylXbFLQb+PQPjk6pnEkZbWCRp9x/qGEvGoH7kGbBeMwjDT0qj3EZn1NWXgVLSWRYgcDgl8cUVR
MySRjOSjbVJIIwQDyPWq8niDUUBk6Ot9R0dLeynlijLLLRR00ZVQAhe2lOs3AbsXC4Gz5ISAcMcZqK81rVL

+z1GbRLZzBDa5hleBleWUnnyw2MgKD1HLEehrB1mOyvdLX+y4dZ1C6S5gllecXDmJVlVmO
1+A2B91RnrxQ2Br2epxaTc+J7mRHlb+04444o/vSO0MIVR7kkVpprF/b3ltDqumxW0V0/
lxzQ3PmhXIyFfKrgnBAxkZ4zyK5vU9Iub6HWpfsN1LGusQXawpuieeJYYg2w8HON3Qjlcd
akg03QbvUrBNLsdRuZI7hZZHuZ7sR24T5ssHbBbIAC8+/AouwN5dav7ye6/
svSkuLa2kaJpprnyvNdeGEY2tnBBXJKjINVvBdwLuy1S4EckfmanOdkgwynIyCPUdKZpWp
R6DBNpV/BdpLFcSmBkt3kFwjOzqVKg5OGwQecip/
CCXa2WoS3lnJaST6hNMIpByFYgj2P1HGc01uBVvdMsNT+IIj1CxtrtE0oMq3ESyAHzTyAR
TNbsrTwolvrWlQi0SO4iiubeH5YpYncIfkHAYFgQQAeMd6nvbj+zvHAvJ7a9a3bTREJLez
lmG/
zCcHYpwcetN1NpfFT22nQWN3Fp4njnurm5haEMqMGCKrgMSSBzgADNLv3AvvrN7c39zbaT
p0d0lq3lzTz3PlJvwCUXCsWIBGeAOcZrH8Qa9dXfg67ns7KWKeOTybmN5Qj27hl7j72cjB
HUMDVnT7yPw1c6jZ6hHcJHNeSXNvOkDyJKJDu25UHDAkjB6jGM1Uu7W+vfDHiG8+wzpLfz
iaG1K/
vNiLGoyv94hCcdeQKG3YDrbOW4mtUkurYW8xzuiEgfbzxyOvGD+Nc5r9vpdz4w0hNWhs5b
cWV0Qt2qsm7fDj73GcZrTuNM03xJBBdTx3oVQwQGSe1Yc4OUBU9u4+nWql7pEM/
ijSElshcWcFjcJmZPMVW3Q7cls8kA9eTg03sBl38GgWOq6N/YC2dvqMl4i+XYbV8yH/
lpvVOCoXJyRwQK1LfxPJcTX7fYFistPmljuruWfaqhCclQFJY4AJHAGepNbNtp9lZMWtbO
3gLcExRKufyFc4NGudR8LeItNKtBLeXN2IjIpUHcx2n6Hjn0os1sBZbxFqUWnf2rNoZTTQ
vmMRcZuFj67zHtx05wGJ9s1bu9bf7dBY6XbJe3UsP2g7pvLjjiPCszYY8nIAAPQ+lZ9x4k
+0aNJaw6deHV5ITH9ha2cbXIxy2NuzP8WcYqC2gbwrq0U92sslLpsFq9xFGziKSHd94AE
hWD8Hpkc9qLgQNqlwvjsSXtg1vNaaPcOyCQOki+ZEQUbAyOCOQDx06V0k+sJB4Zl1owkol
mbvys8kBN+3P6Vzs5udb8VyTW1jcpZnR7i3juJoWjV5GaM45AIHpnGcNjgZqG61Zrr4fXG
k21jePqn9mm2ltTbupjby9rEkjHHJGDzxjOaSdrgdHe6xOmox6dp1iLu7MQmk8yXyo4UJI
BZsMckg4AB6HpWL4g1zUX8NavEumvBe2w2y4nwqoRkSI+BuHGOgOQauSTf2F4iub26hnNj
fW8QM8cbOIpI9w2sFBIBDAg9Mg+1R6ndXuueHtcFvZS/
ZTDstA0bLLOQCWIU846AcAnB7EUNgdFZy3E1qkl1bC3mOd0QkD7eTjkdeMGp6htLqO9tku
IlmVHzgTQtEwwccqwBHTuKmqwCiiigArm/iB/yIOs/9e5/mK6Sub+IH/Ig6z/
17n+YpS2YGb8IP+RJsvrJ/6Matsf8AJQD/ALg/9AFYnwg/5Emy+sn/
AKMatocfEBv9wf8AoArF/ChHWeI7qCy8N3NzcyrFBEoZ3Y8Y8KARXkPgWK8+JXxYt/
EhtzFo+iIqRlh1Kg7Fz/AHizFz6Dj0qDxz4lvviTrv8Awi3h12/
sa0YNeXS52uQcbie6A9B3PPoR7b4I0ay0DwxZ6fYRePB5yfYRCOFEBPq7HksT3JNZPcDpKKKz01eB/
EM2jBJPtEVrHdF/F8DaVd3UDrnOUPbuKQGhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXO+M/
BWk+OdGGnaorLsYvBcRHH8yGeEgSRN7Eg8EcEHr9QCOiooA8MP7N1sCRH4qu0Qk4U2gJA9M+YM/
AAzfD/0Nl3/4Cj/4uvc6KAPDP+Gb4f8AobLv/wABR/8AF0f8M3w/9DZd/wCAo/8Ai69zooA8M/
4Zvh/6Gy7/APAUf/F1hav8GdJ0XXtE0y48SalO9hkbWJmBvG6oRz8EsWnRC5Uo7QDkvbxabBSBUHOuXHUzTC9IMEq+OEW0nqhxAVvJpdbKeQGXrpIXW23BEHnxESVi+ulspT8sY5MRJRzfyaf36KSx/bT80dMpb6YqwoR8hpjqeE/hg8i/SdNNJ0ah8U1RfMHHxXkEL/
wzfD/ANDZd/8AgKP/AIuj/hm+H/obLv8A8BR/8XXsOgar/
beg2WotC8Ek0QMsEilWikHDoQQDkMCPwrSoA8M/4Zvh/wChsu//AAFH/wCLo/4Zvh/6Gy7/APAUf/F0Nl3/AOAo/8Ai69zooA8M/4Zvh/6Gy7/APAUf/F1hav8GdJ0XXtE0y48SalO9hkbWJmBvG6oRz8EsWnRC5Uo7QDkvbxabBSBUHOuXHUzTC9IMEq+OEW0nqhxAVvJpdbKeQGXrpIXW23BEHnxESVi+ulspT8sY5MRJRzfyaf36KSx/bT80dMpb6YqwoR8hpjqeE/hg8i/SdNNJ0ah8U1RfMHHxXkEL/
0Nl3/AOAo/wDi69zooA8M/wCGb4f+hsu//AUf/F0f8M3w/wDQ2Xf/AICj/wCLr3OigDwz/hm+H/obLv8A8BR/8XR/wzfD/wBDZd/+Ao/+Lr3O

1aqPmyemBRzIDp6KybDXory7FpPZ3lhcOC0Ud3GF80DqVKkg49M59qpnxjZvbm5tbDUbu2
QsJZoIAVi29c5IJ+ig0XQHRUVy2p+KGt9Z0eO1hvJ7S5R5GMFvvEylMrtPXjqcVprqNlZS
6zc3F5KsVtIhm845SL92hwg9CCDj1JougNaisS38TQSXMMNzYahYidxHBLdwhUlY9FBBO0
nsGxTrrxJbQXdzZ29peX15bMolgtYwWUFQwJLEADDdzzg4zg0XQGzRVPTNTttXs/
tNsXCh2jdJEKvG6nDKwPQg1cpgFFFFABRRRQAUUUUAFFFFABXN/ED/kQdZ/69z/
MV0lc38QP+RB1n/r3P8xSlswM34Qf8iTZfWT/
ANGNXLfELWb3UfGcvhzw8TLe3O2CVoyOMqNyZ71l6R40l8N/
DvT7HTGMmr3nmLCiDcYwZWG7HcnoB6/StPwT4R1XQ/
GEUdxOi6nNFvlYsSV3LuKk45PPJ9a52/
dSEdRpPgrxT4J8MS2VjdeHis8gaSWS3mmMsjZ4BbcBgDoMevcmux0a2+JItNt74XSLYMbr
admxjj+MVt2vhu5nkjk1K7EqL0RSTn/
CumRAihVGAKhgZmiJr6RTf29cabNISPKNjC8YA77t7Nn8K5q/
0ltX+KtzFJe3UFmuiQGWK1maFpT5820GRSGAHPCkZ4ycZB7qsuPR9niq41vz8+dZRWnk7O
mx3fduz334xjt1pAZXh2OXTPFWtaIt1cz2MNva3Vutz08zxGQyqy73JYr+6BAJOMmuprNt
9J8jxJf6v5+77XawW/
lbMbfKaVs5zznzemONvfPGlQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANkkSKMvI
6oi8lmOAKq/2vpv8A0EbT/v8AL/jXhnja3vviN8an8G3V/
Ja6Pp0Qk2RfxiX2gnoB065v+Gd/Cv/AEE9Z/7/+xf8Axusp1oQdmNJs9d/
tfTf+gjaf9/l/xo/tfTf+gjaf9/l/xryL/hnfwr/0E9Z/7/+xf/G6P+Gd/Cv8A0E9Z/wC/
sX/xuo+s0x8rPXf7X03/AKCNp/3+X/GvFviXe2k/xk8MSxXUEkaR2+51kBA/fX4U73WQEL++7ntVz/
hnfwr/wBBPWf+/sX/AMbo/wCGd/Cv/QT1n/v7F/8AG6PrNMOViAG6PrNMOViANBG0/7/AC/40f2v
40f2vpv8A0EbT/v8AL/jXkX/hnfwr/ADO/hX/oJ6z/39i/8Ajdf/39i/8AjdH/AKjdH/39i/8Aj
jdH1mmHKz13+19N/6Kz13+19N/wCgjaf9/l/xo/tfTf+gjaf9/l/xryL/hnfwr/0E9Z/7/
8AG6P+Gd/Cv/QT1n/v7F/8bo+s0w5Wa3uIpgOjcNj8qmr5o8PhLHp/
ijwtrN5ofPhQHzeZhgDEyzqqXO1yAqfv8y4g3Ybx4bQEDCzBQsUnkL44pHXv0vKiiyN+/
WtoTU1dEtWPaBwY3iKOksn0/15P/6HBXtleJ+J+v+TlvF//AF5R/wDoMFe2UjO/D
o2Ovc68L+O5Fj4tcPtCtpDcEyAZZ27XjY/PTn4Ip1dgwoooooAKKKKKACiiigAooooAKKKKAOX1aK71r
xRb2VnefZ0+MtNDZtP+8r5WNcHjhd5/
Far3Vre6F4h07Wb3VGu4u4ZiNPnzCse1XOUb5euHwOem4114RVZmCgM33iB1+tDorrtdQw
IzS5QOY0K8mPEOni4iN6iL6aQ2+8eZsvtvxHI56c1z1te20nhPwjdx6ube2sFK8ntj
G5tnMJUFwysF5O05HG7tXo/
lRiUybF8F8wjWG7HOPTNIIIYgHAiQB+WAUfN9fWlygB+WAUfN9fWlygclstbjxDpMQ1+
+1WeKRrhUQ22yEbGXe5SMHad20Pl4MRR4UtQFABaUkY9ZGzWxbWVrZhha20MAY5byoV
Rhc/
XFSqqooVVVVZ0gdMU0tbgcFYXxlgfe/VVZAVJBLm9uYra3Fptmzdx0CkBMAwCTwOlWNV
bS4nRV3Fo0jhLcd8fe/
4DXZtBE8QGiaJDGMMYQqMZ0kAnkSDkrS5QQW8QqHnGdqIkhqa6+AgtkA9FSWN+V4Lxd7w
bS4nRV3Fo0jhLcd8fe/
4DXZtBE8QiaJDGMMYQqMDD8KcFUEkQqkMGfDDR7KO5KEcJjaTBPmGRhYRVdLR0BbQZdKt
fx0CgbifatDR1Ua74hbA3G7iBOOSBbxf4mtOCxtLaR5ILWCKSTl3jjClv94ILWCKR/
vNHGFLfUgc1MFUEkKATyTjrS5Rkt+pL/
AOgpaXFtBuKEKt4qbjmwHbuGSRaubjl8Ryupc0C5sLDUGsbiEqqow++Vjp5hyF7xe6wqqow
oAHtXNfSwgUYBRLUEkAAtySB1pcR7J9VkL2+vCArlHIHnAGnDMlY5IYOMhsKFDBbgmIeF
KZEiaD/8nTa3/16j/0vfB8oDRMPeZ152K/

iGkdjA8W6vc6Vp1otpLHBPfXkVmtzIu5YN5OXI6EgDAB4yRWfdaT4k0SEX+m67f6zLGwMu
n3qwbZ1JG4IyohRgMkckcYIrR8UX2m28FlYaxZJc6fqdwLSVpQDHGSpZd2fUqAPciud8Sa
FF4P8PXetaDq99ppsYjLHZvctNbTEciMxuTjd0+Ug81nHZIbOt1fxDpehLEdRuvLeXPlxJ
G0sj464RAWIHcgcUQeItHudHk1aPUIfsMWRLMx2iMjqGBwVPTgjPNYOhSiX4g6zJex+VfS
2Fo1uj9RDht4X6SZz+FV9c1HR4tSmtrPTkuNSudTtYHeWRkgNyELoWIJyUVASAOTsH0ORX
sFzf0vxboms3n2Ozu3+07SyxT28kLOB1KiRV3AeozVfSteSPRtRv9YvY44bfULqESyYUKi
TMiLx1OAAO59zWXqz3sfjXwhDf6hZyzNdTssNvbmM7fss2SSXbIzgdv8MOb7Tv0ry7m0t4
f+Em1DMl3AZYhLvm8rKh05znHzfe298U1Bf18wud5pHibSNckkisLotNGNzQyxPDIF/
vbHAJHvjFa1cdNa348Y6C2q6zpsl0hnaGG00ySOSRPLIcMxmfCZKHp94KK7Gs5JLYaPKP2
g/8Aknlt/wBhKP8A9Akr0vwn/wAibof/AGD7f/0WteaftB/8k8tv+wlH/wCgSV6X4T/
5E3Q/+wfb/wDota9DC/wzOW5sUUUV0EhRRRQAV4n4K/5OW8X/APXk/
wD6HBXtleJ+Cv8Ak5bxf/15P/
6HBQB7ZRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFA
BRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVieKvCml+MdEk0rVoi8LEMjocPEw6Mp7
H/ABrbooA8H/4UR4msSYNL8cTxgWPyJ+8jwPor4o/4Un42/wCh+n/7+zf/
ABVe8UU7sDwf/hSfjb/ofp/+/s3/AMVR/wAKT8bf9D9P/wB/
Zv8A4qveKbI6xxO7MFVQSSTgAUXYHzLqPgjxDpviaHQJfiBcG+ltzcDpviaHQJfiBcG+ltzcBRLNgLuwP4up5P4G
r2r/C/
xnpOiT6q3ji4lhiVTtWeYE5IH973TYzE5IH973rhNf8VahqHxNk8RiCYGzeZbx7SC1uoIGPYoCc9OSa+
iNZujd/C67mB3RyRujeql1INF2B5hoXwy8ZB5iSRc4aeYkc/
71PPwv8ZDV30/hN7rcv8fnTY6A/3vevWvh5/yLtp/uf1NSP/AMjnq/
6CKLsDzUfBTXP8/hB/Zv8A4qj/AIqXSizDwf/hSfjb/ofp/+/s3/AMVU/Ck/G3/Q/T/wDf2b/4qvcLsDwf/
hSfjb/ofp/+/s3/AMVR/wAKT8bf9D9P/AN/Zv/iqveKKLsDwf/hSfjb/ofp/+/s3/AMVR/wAKT8bf9D9P/wB/
Zv/
iq94ouwPmrxP8AD7xJ4Tsb5iYbvUviDOqTTpboPNm5Zjj+90Ayt7A1oWfwk8Z30mPvV6x
5mm4H/fVZf7Qevz6h4n+z3wDf7P/8Ad4fVW2eVP/
Aq6+25+LepfQ/ySu58VeJJ9UhiORAC3zexZmJA9hj6163on/IPh/65r/
iq94oouwPmrxP8AD7xJ4Tsb5iYbvUviDOqTTpboPNm5Zjj+90Ayt7A1oWfwk8Z30mPvV6x
5mm4H/fVZf7Qevz6h4kg0mJW+x6cB5rgHaZnGdpTITB/
E16Z8IdYuNV8LxNcK6zQkwTqQ6+vuRg/jRdgeWaB4H8Ya/c3mh/3cMEPjG7iaBwpL3Epzkn/
a9q1tT+FfjPTprpPTGt1fxxcv5xIGJhjrGP9r3rsvhv/AMhfqv82rsfFv8ArtO/
32/9louwPKovBhvRfWNG3v0tqAwwOsvN/2s3/s/
AMKT8bf9D9P/AN/Zv/AAN/ZF/iqP+Kt/Q/T/wB/Zv8A4qv/wDwpPxt/0P0/8A/Zv8A9/
iqP+Kk/G3/Q/T/8Af2b/AOKr3ii7A8H/AOFJ+Nv+h+n/AO/s3/xVe8UUXYH8F/wCFJ+Nv+h+n/
wDFV7xRRdgeD/8ACgdA/7+zf/FVXeqv7+/Zj3/7Fo3/ABD9i4Ztmv9x0an/
C3xZ4f0f9i0vVh9dWbrYdAsIfC+Is1NnCSdHrmeSN/xoeqGLaPrRvj7VHexIAPm4GqMo2Ro
of2h9flh0W10C2V2M5FzdMoHtiViVfgPrFzPpz6dMGDWuCgYYLRNkqfcZDD8q
LsDl08D+L38T8t1/I0V+Vfh6A7DgnGXK8hjQxjqOR6rQY9ulateKf8K5+Mf/4HHuuYn/
vVp6Z8A7i41KK58Ve+VeJJ9UhiORAC3zexZmJA9hj6163on/IPh/65r/+kD0vTo/
9jbD8p7E5Ub+47u0WZwRSj81Rz0d+CTMjmqgkmqBLMNZyIYvGsj2TG4r0GJHOcqgJ9X
IlyTnuQ0Kb1vkl8r/NQ+G+g9zs3S7DnT6KXCdL19PEzxdzKtvEz4A/ZmeMtwgmNjjwnMpmq
+13Ag3EuRHOsHXbE3kXPjfFs7aojLYAgjJcR4UhYMvPyxD5QdpPnOm3eXKHaC/KbqZDpfv
4sAbF5M48CcFVGS5eprQg/PH0xt6aXDvI5xRcHakjnfQzO9S0f/cpoPjJ2hsEUtPWmZ7ZZf
Uw5NpBDTgI0t3DaFHlPuKQKWlvCpWE6lxjXszCI5dC5/Ss7LnBEGsdZfgFyUzdPJNO/
2mMpjeiFoPxqKVpeuxCKTp7Ql3tBPfzuFzFF+hNwEtmI7nCD/5BOvz+2C5A0+x/Vy6t+2J
Grv+GbYn5rlLnRpy02DgxqOkUzmpRc5cobdajKGBaLVSCsZPQ/eeTrfhwbg6zT3jELwiXqY
WDjutbAQR3SfPq8PjM/Y9gsUfcnFcMFHtrDMgHIxvuH3gG9HiVfOsFzZZc0cJXnMb7
rZotIqutSrJ0ez/hlT0RMVlUjtxJAiInpvTcsfCAa/1ku3G/HnTVZuxJdDk6pvrDrv0zSx
Tgtvf4E/+CQgb5ZeLM9eSoGb9vl7TwEovCljnrzKwwm0afzbZw5ZmnljDm6Gw8K8GUW1X0V
Wq23ojAt/c1cLrVWpZ1ccHJLpR2QbNEzcv3n7HCivBLQ40Flxtnpg3u6bu8xuZxU6yg
1G+0SWvbfLRt4YqKXy9fGhM+1EZsWyLpDIT6S1Kh4/Cy4gSaqCpsmFsBzU7d8YTFWo
D9vS8NUYrKc5n7I0h7EtvFBFQDvXMehRTuxv0qYQsnypkSoQJsZ0Q9wdMvupEjG/
NgL3uNcXHoDdqAE1tyg8Zt2OzrbCIt6l9W5rp6uLCWbvATk/

8Axqj/AIVz8Y/+h3tf/A6f/wCNUPCze7HzI0v2hHRfh/aIWAZtRj2rnk4jkzXp3hVSvg/
RFYEMLCAEEcg+WteTad8E/
EGt6tBdePvEv9o21scx28E0km8dSCzBdgOBnAyfUda9vVVRAqgKqjAAHAFdVKnyR5WQ3di
0UUVoIKKKKACvE/BX/Jy3i/8A68n/APQ4K9srxPwV/wAnLeL/APryf/
0OCgD2yiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKA
CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAAnAyaqXF3bbWil2MrAqytg
gg9QRWf4k1J7KyRITiWVtoI7DvVCHwvG8Ye6uZDM3LbccH8etAHjfi6eI/
tD2MibRGIFHHT/
VPXqPiy7tl+Hd5DFsUCNAFXAA+da8p8U6NbxfH3T7FZZDHJbhixIyP3b+3tXqGt+ErKXwr
OpuJxlV6Ff7w9qAJfh/qEMfh+1Bdc7PX3NT/
AGuNvGM7gjbjr2+6KpeGPCFjFpcIFxP93uV9fpW9/wAIrZf8/E35j/
CgDXXU7cKPnX86d/adv/fX86xv+EVsv+fib8x/hR/witl/z8TfmP8ACgDZ/tO3/
vr+dOTUYHOAwP0NYn/CKWf/AD3n/
Mf4VFc+GVghaa0uJPNQbgGxz+IoA6lHVxlTTqwPDt891b/
Ocsp2k+tb9ABVeW8ihOGcZ50/TZJkx5hwq/
U1hWegfboFur25kMko3ADHQ9Mk0AeaftETW8nh7SUkE3S5YpZHevLPj5e1eieH/C9m2mJmeb7o7j/
CgDnPh1exR6t9/FF9FLNp+1gcO2cf8BrmPC3g2wivr/z8TfmP8KALLLuzV/rizFzc4t8xL4/
hTZfCkIjJguJBIPu7sYz+FAH
Sxyr0IPlNPrldAvZjdsM3rhkZ+tb9IdB6E0W8ihOGcZ50/TZJkx5hwq/
+vVf4QXFtF4Q01nCeasG3czZ0fUHxf0C1svhlqlqwKhCMM5ZNbOFFGxz+Ua0HbW68H6
dM80qs8AYgEY/lQAtvexD4rfCAwHB/uS/7/wDSo/
Za7fxLfSaIqqwJ8xeAfrXIR+DbZapaNmC7hWZPYEZwfcdD9KvVHBbw+0jvxUSZzZKA
GdR+7Xv7of2nb/b/AL0/4Z3xbgxS+D11Zm+97mMXqAcmvlPIdl81g9fdiwmUmvKVzDcM
ERKUVdTjVP0NDU6a1Vkpp9H/aR9m+ZwIk0NDNOR4aDJbrsYDXmTj+C36MyB3q8m6i/vI
JTJLGX2+HNNSC2CdQg0KiP3IQd+O3tHr4ERl9pd2XHNYmh5vlUsGLKEVOeeA/8iFoNXY
xXoDrbKuWMVmT7VvtuMOUhqQ0o7anaSx2bQhbwgPhO43GuZzJuEfyypiiiJ3wbZDoLCu
OOedAUtdtxtK8zKLiNGUXVBQpHdRRROdqxPZ9wuD0OJMH5K1JlbVZoh2OS2CKvkcBspB
7WMzx8Uy/EQ7HvNUPdGvDS4zv0RuAH/wBQ3kRHLD5JA4kxLhrOlAkcX3ayGd+HNIJFIO
0rWtNf8TrZQKdDugRGox5Enp9K1Lr4haA8WBqlkef+fhP8acnxD0ARqP7UsuAP3+PV1
kvxk1PV77Q9Oj1HT5baNklWeNlydp45rudG13xGtigj0a4YYHS5Fr/wDCuR+NviXT/

EGg6bDYXUFw8dyWZYZA5A2kZ4r0HSPHOhWlkkb6pZZAH/
Lwn+NAGdYXfiKwkldNGu2MhycwPx19ver/APwkHif/AKAlz/34k/wrS/4WJoH/AEFLL/
wIT/Gj/hYmgf8AQUsv/AhP8aAM7/hIPE//AEBLn/vxJ/hR/wAJB4n/AOgJc/8AfiT/
AArTj+IOgyOEXU7NmJwAJ1JP610NnqUF4uY2BoA5Kw8ZXUd8trqdrJayMRjeCPzB6V3dtM
J4g4rjfiJBG2iW85UeYk4UN3wVOR+g/Kui8POz6XCzHJKKT+VAGZpH/Ievf99v/Qq0/E/
iW08NaaLi4JZ3O2ONfvOf8PeuYGv6fousXkt7dQwKZGAMjhR94+tcd4t8ZaJrHjXRS19ay
2UW3zP3ylfv5OefQCgZpXvjDxVqVjK9v4fuWglQ7WSGRwQR6gYNP0TW/
FEOlRI2h3YILcGCQdz7V0MvxD0AxMBqllyP+fhP8ajg+IWgrCAdUsv/AAIT/
GgVzhfibrGu3XgHUYb3S54LdjFukeJwB+8Ujkj+JwB+8UjkjHXFJ8PdY1238L2Mdppc80SxAKyxOQR+A
q98U/GGk6x8O9SsbO+tpp5DFtSOVVWY4lQnAB9BT/
h14t0jSfCOmwXOoWsqwAMjzKpB9wTQBeS78RJqUl9/Y12XcYK+Q0t7Vf8A+Eg8T/
8AQEuf+/En+FaX/CxNA/6Cll/4EJ/jR/wsTQP+gpZf+BCf40AZ3/CQeJ/+gJc/8Af
9+JP8KP+ImgZ/AEBLn/vxJ/hR/wAJB4n/AOgJc/8AfiT/CQeJ/+gJc/
wCQpZf+BCf40AZ3/hIPE//AEBLn/vxJ/hR/wAJB4n/AOgJc/
8AfiT/ClyKaBD7XVXjS8oFM8ZVQ0vLEmGlQnZvIGK6GfnjYKaGGDjUUn1b1A1eeeiN2sXCMV1ZcGOA7hX/
0jNqZEjk48X7jgTWOYCXDlJmSQxOUO7Bbi+fEcHMoGzggkWYDL8+Q9SVJ7mPyaqfCQLe2TCWZZJt+8J/
2TEbPfFnQbuVOeVtiYmrvNN7kPmQYkXHPA2Sj2ZAe1f3gQgZWWMbhEQbCbYJKiOHHP/
gjz8wvExEkC2cX4NkHbtUwMaSB4tmaHgEOkVBb7Do4y7Yb/+AStnKpAwgs0v7TjiyYbtcTxDtmXdz/
f0eOOMaSvKGn8ZCGOkTlNrd6IdG7jT7c9iVJAlmTmGU0Ar/hUaTfxW3A5U0ME0ZSjRSRaHS5q1vsB/
f3/+AEFLL/4EJ/jR/wAJB4nf8cEZ4N+H4ggAtMnFRcSQ4/wA5BR2f1L2EhgKjEwmnvsl+KJbb9AAr/
Kz8fT1AZD3HfmlBUB/Sj/gK/1XbMu5VAGXEdG1SJM/Y1XdRF6UeHK6RD3j7yjx7mG3yWVqlAz4aJfXdE/
wAKB4n/AOgJc/8AfiT/CIPBn93UP/Aj/
ANavQNTnnt7LzLaS2SXzI13LEJtLqGGR3IJA98U291rStMRnv8AU7086xgNgHBy/rDrRC3kl/9aVXfTfY6/
+2uoVmibGMqwBH6GvGir/wAK/eUuitYh6GAVL/7Ffp79oWG/
wBPxtPqNnkoCp4dGMT/l8/LPdRp5npnm2BUOtK9cyCDuVgo/qeFEJHm7B/wAo2/qL/XAt5ydzDUMq/wDFA/
2dYF4JKa7sVG+8SWvhQ4OMO/Bhat/DDu8cWpqLQJmHX59/AcHn1Ab7HAjGXnhUdNGUSHZ1qIbHjR8uZcHNm+/
SBBa27AH7pLNJd0ZaxD4+o+/+gnpIWOKh1MYU/+S8GjG3LC+hrfjhVUpm+0IrOSIb6Ey/XhPGvMSVi6W/
c3TtwMpt4b77yQQ2PrjyGsCY8DJt8Q3Z7qSfjkwpT7/OUqnTRx9DBUuLThGvzN+JmsQJsbt0vg3iiMv9wXc+/
J/
jzE1bNk+4J5v8X0aeAx8NrbUUhUN6WsvX+N9Rh3/2ZhzuyOAOlp9LUSrRAMTqLIYs/3r1cbFJxKJeh9uEq/
UhSaU9ulYxGIFc3bjgZ7f5oydlT8Lq9kL+ZsnGMHgS9RJKx/eOBAPsUBjDNU5FEomW+ggD0JlJeXLF3y14/
AQAzR1Z3pGv5zS3z9ZugGjWaASrtv6Q4XKaVHDKcg+NAVd6tE/X1mF7Ng/Wf5u4kR8tKEhAI9ciCLwtnB0/
a6lYDf6xO+j99ohySOYqGr5hQ9/xG8pSbPSBfb/T8gk0oZXDNIDgJNygACk0GgJd/vwmR+PDl9ytQFRZmK/
dJ/+h+GjPmHkXSJ+lI0fUOQvrCP4n0W4ISqYZeLdY+3Bg+6JEgBm40CCQvNvqaCEoL7CWCF7qxcuAEeJ4pQH/
CCRqHQUaoQ==

wCtXollqFnqVsLmwu4LqAnAlgkDqT9QcVXt9e0e8vWsrbVrGe7XO6CK4RpBjrlQc0AcH/
wojwZ/d1D/AMCP/rUf8KI8Gf3dQ/8AAj/61d5f67o+lSpFqOq2NnI/
3EuLhIy30DEZqzJfWcKwtLdQIs/
+qLSACTClvl55+UE8dgTQFzzW5+AvhGW3dIJNQglI+WQTBtp+hHNc58ML/
U9F8V6p4Rv5zN/Z7kROT/
CGxx7HKkDtmvbbO9tNQtlubK6huYGJAlhkDqSDg8jjggivD9DcR/HvxKTn+L/
0JKAPTviAc+GoT/08L/6C1bOizLbeH1nf7scAc/QLmua+IuowW/hWF3D4+0oOB/
stTD4rtIfA120UUryLYsVVgAM7D1OaBo8u8L+EB8S/
FGs6tr13cfZYruSKOGJgDnOduSOFAI6DnP596PgT4Mx93UP/AAI/
+tWZ8A0a58M6peSHLvqL5+uxCf516H4o1m40SxsprZInaLMvqL5YSAkBJJArEYI5weKBHH/
8ACiPBn93UP/Aj/wCtR/wojwZ/d1D/AMCP/
rV6ZWdZahiwWXULqxWRnlG6GT92VVm6EnqFHzehBoC5wn/CiPBn93UP/Aj/AOtR/
wAKI8Gf3dQ/8CP/AK1d/
p+saXqwc6bqVneiM4c206ybT77ScUyXXtHgv1sJtVsY7ZcSJtWsY71jgW73KCQn/AHSc0Bc4P/
hRHgz+7qH/AIEf/Wo/4UR4M/u6h/4Ef/Wr0HUNV07SYRNqV/a2URO0PczLGpP1YRNqV/
a2cTHAe5mWME+mWIpU1OwksBsWkVvkvrZrNiAtwsqmM5OBhs46nH1oC554Bxfr+jJqAPXdZcyeEb5j1Nu38qoeAP
1q4iwsrz4Z/FKDw3HeSXOmXyCWHzOBK83Dx8QR6ODcob0DDjpncpBx1HNe+W1/
Z3rzJa3cE7wNslWKQMY29GweD7GvFPiYQvxy8NE9N8DHe+BPE9BZxf+jJqAPXdZcyeEb5j1Nu38qoeAP
+QLH/ALzfzNP1m+ii8F37sBwtsxB7/sGwtsxOB7Vh+BPE1jHoyDZMSGbjaPU+9Azpbz/kb4v8Ad/
8AZTXVp9wVy+sNPm+ii8F37sBwtsxOB7Vh+BPE1jHoyDZMSGbjaPU+9Azpbz/kb4v8Ad/
8AZTXVp9wVy+lQXGqas2pzRmOMDEY9a2pzRmOMDEY9eMfyrqgMDFAwooooAKKKKACiiigAooooAKKKKACi
iigAooooAKKKKAPnfwn/AMlp8bf9d5v/AEdXpleZ+E/+S0+Nv+u83/
o6vTK6aXwjCuW8ZafBq02h6fdLugubswyD/ZZCSDoo21hLd40NUhQIpZ0s1OytaswQz2suoseXyhYmz+MeDiG
FInH4sKtq6Ay4dfuLVr3TTYrPL1s9gQ/
Q1S0OzuNEh16z05YnudNjLKLQ8xwu4wx75G5wx4G5GerMfWr8nh26k8Y6rNdTwQ0QVRCPp9WlvK+EnSNEDI/BwCVPY9Bmp1Ahi1efTwpM
93bg/VRUeq6Bf3J1aSFIJRPeW9wlvK+EnSNEDI/BwCVPY9Bmp1Ahi1efT9a023/
AOEhh1eO+naCSLSLZEGhOxmDKUxlcJ0uLuwu8mCPBznjwAwM
e1Je2mvapdaVcf2dbWkOn3wSzfZ2uAzSDayHlVwuA3A5z7Y50be01G217UyIIHsL0rKJvOI
dGESptKbeR8uc570Ac94EVNJi0y2VdsOq6q6bDdJjp5yIqyfiVKH/
gJpbnO0o+KLTV3UeWmqrY2rf7Eccu2Y5zH/
ABV+XQNUi8Ui8aRaWaUk+MesabHF5Tsx2bguxxn0Klv0q6dAa2sNBd+uwxXKwLjXTgXXSHSqbo+9
tn8TRZ2sBe0W8mvrS0hI7ILJd3EIwMfkKrKv6AVhWd/
PqTeGbu4KtOwAYRDdy0ZMmqrY2rf7Eccu2Y5zH/ABV+PtTAGsG/3XwC8xTCmSr/A434q/8k51P/
+jUr0b4cf8k28O/9g+H/0ANBFec/FX/knOp/
wDoIrnrfEI6Z2iv/AIhXXUP/AE8L/wCgtXcXA0O5Zk1hYpp/gr+Bn/3/6h/mB3L/Xb8m3/+PfXvVVdfcd/
r3j/wDQDQpBxkBhSPTOefWiKrFB7+/8AdY+X/6D/
8AXrKa1i/EPwaR1yHy+fw8Gu/Efm+kmXPXxF/7xXpa/3rqNZN0J3TQQv7o36Sit43V+gQ2fZ/
gr9ZqzOfoubS0o2Lpu3+Ga2fRM+3UvpS3+J3Bn70jvVzUf6xdGFyckRlVtz6g7N5QOPfa/8SvUc0K0fl
nHzdeD8oxjGMcBY+F7Cyvo70c2bSnPQnyf3Xvwkv4P7PZUVs/Ys6kDvstUzOr0W70CtiWZ2lw/j3H0AC
KLRbu0nuvtV/
c7p5g4L+YyKmMnORl8jjpipbazg1XxZ4Pa+QT7NBkn2yDcGcGDBIPXBOfqAe1dFD4I0aFf
JAu2sQcrYvdyNbqc54jJxjPODwOwFX7Lw/
pmny2ctvA4ks7draB3neQpExBK5ZjnlV65wBgYFAGnXhHwYUHxD4hyAf8ASh1+r17vXzX8

OdE1DWfEeuJY6pLY7Ln5jGWG7LN6EelAI9N+JZgu9a0KxhCPcIzFkAyRuKAZ/
wC+TUnxfs4rb4P6htiRG3wchQD/AK1a6Pw38PLXRr3+0Lu6e+vTyJJBgKT1IGTz7k1l/
HMAfCfUgOgkg/8ARq0FFv4bf8k30D/rzSmax/yU3wx/1533/tGn/Db/AJJvoH/
Xmlauq+HdO1m6trq7S4Fxaq6wy293LAyh8bhmNlJztHX0oJIvF/
8AyJWvf9g64/8ARbVk2Njb3HxA+1TRh5LfRb1HxyxyRp+M1SoK5dmiF6aMHxHMkZ2B7+uNZc7dGfl/ztVAP9FG9Ty0P6FqM/
8M6ix/ARyY8iP6WkqLG+Fw3vUBzIUZKd6sS/PGxYMHJhxqHJq8s5sJQWFe3m5kVWc
tPkHUwTrcj3ReJpsckFJdF+UEcI6D2CQMhW/uqIgKb23qwxS/
kl0658f/
2eRAX0+zlYodgDOZEd8jodgGT14qbW9O8RR+GI4hYeFdOt7IxyWlzHeTH7MwI2sv7nucD3
zjvXeHR9Pa7vbp7ZHlvYVgud5LLJGucDzZUvZGsDOe35uwmVmc20u0o48oN7Ew6kiH4mUMtDEVLcEwkpp2T5yNYUmz7LNdJ6Rf
sv3SqE447ZzjtigDN0+We71HWLjRNKtZ0munhu7zULKQXeMBCiKsbEopXABK9z3zXLaRaw
6p4Z8DWt2I5rc6zdoUA+RkT7VtXH93CgY9OK7ybwfpct3cTK15DHdOZLi2gu5I4ZmPUsgO
MnvjGe+asWfhfRrBbZbaz8tLW5e7gQSuVikcMGKgnABDt8v3eScZoA1lVUUKqhVUYAAwAK
8L0AA/H3xLkZ4b/
0JK91r51XTbzVfjl4it7K+ks5NzMZEJBIynHBrQCPUPijPbL4ctLM7DPLcK6p32hTk/
mRWtcaclp8M9SDQIsq6XKCdoBB8o1Bonw1itr+PUdV1CXUbhcMocEocEAEdCckk4ro/
FqBfBOuKo4Gnz/wDotqCjy39nv/kRtQ/7CT/+io6674gf8gnSv+wzY/8Ao9a5H9nv/
kRtQ/7CT/
8AoqOvTNW0ex1yzTzFpqETyRLIsq7JXjZZXU5VgyEEEHng0E9S9XmtnY2+ox/EJ8+y/
2tqDmNvusVacjI7j7Vl6eFNOjkV1udYJgdjdrN2w/
EGXB+hq3DoenW5tDfsks853t16Z4oA57XE+y/
EPRLm0iUXMQi2FsWVgFoZuJ0rbAmFQdGrDKSLIBkRoaZxxFr1a7W7tLfZQ8WY9PmxTLxYp8/LwsFb8XLAbfyIDpXvLKU/
0PUzDCJBZFGJnhcYExk52UqB3XOaBd4nXLWdHUI9+NmH0C2ogYnQgO2AU8nQk5YR+BrXdzkG1l9tHR3uk8oR/ovP8JOPVzrb//
AbY//ebb4oB8/Kys4wG4sa+ZgSQpbC4z7CCvEUcnrWgsBoR8T6tPvjHdczr+/CJ0iCbwn8GX4E8nuJpHuxpbzZqSRBIL0P3G/
7nVcg5e9IpEAYNfBQ4+oqwsMCzvHN10LsbwVLsJkXOQyAADnwdgn5wuUMDwA/8fNo3XrrsJi/
uNoAHm2+T3yvnj5B/wGsz9BTJd+w/bCuw1TcIuGf/SJMsbFB52FsWWdsn34XUrjHyvWvsxpYV1Q3XuYIgwKYafaEKAGvA/
wCNFr3d+2WTaSnfAuZB11Bl4Bb3BsWgiWkIT+XAgQGCUKCUgV1gH1i2EQF5L6y3m9AZU9y/
8AmpvuQj71U2PEh4fVmgvBZYUc2KLb4+NIJTN5v2j/AKyBdTGBMnTXW0AirJz/
8Az7WrrQDfpR/lZfzKF/tVUA/AKZxB9IS8+2Pd0X2gPYfYZmu/BUh0/XytVKbW+8RH10b0gdQhwMbDKP9lDIA8H7DxSFVWJdcIkeQF/
iZ3lK8oq0W02S6Xzr1QlNqfoTqC/lhAA4lN1rJ/J1jWiBcTIrfsIx5l5FL2aoEMnTo+b94WKkEKTDn1wHqqwuBG5/
GXdUCy89YiLd/kMnS9U/EH2yQYUxHhr/ZrrzTxa9l35qwd6FK6L6Gt/N73sJ7o5tS9Xzn3XYA/
49vPtG/QcmdEfHdbuK7EKnMCVmsBHVk9/XDTtO07mJp/x/Km+Fvf88pv0gOQ0jFl9sqRqc2JOXyzKnfx7hVk6wz0V0Ts6fbf6tF8QPz8/
+Tq7pqr/Xb7pU9QRbafq9MK19CZP2ibw9Ljt09/
CwvbE8I6ue9XWU4wYXD8HgZ8/RD5trgIXIW2s/qcB9GZhbWejzBkZZyfHSTH0SHIrpwX0m/
NjiMLqKzgwM2HFF7eRzdFXXdHTVJIzCPNh01THsTbVvDmAHNtlg9uThd4QU8MrLVAJSdjT+yxHG7wmzMmypO9jwQ2TLwiL17xusGDC9/
ga7H2iPMPnGT6D4O0NgT7T4U2rpPW1HkMEHq+7bqtbAaHC6wxTj2aYm/mPtDPJkSF/v+AW9BCnuXtbGowOxl7XgHK1/NlkHm9h6Nk3/
iDTOMpZ4P9+HZQ4M3pocGxxbZyhSKO7Dp4NQ7W0gQ4FjzECfGkx6+Xq29oWS4mcVXRmwKlEQCgeK8qdVy63m6TKMmp0HH5sldLSLuTMdEv5/
DOe++V1dQoyuHIDn9sLGaBmQSABHT/zw+ebWEPWGT+jcQ4bcH9E/2ZLvbhmNEDU2cp9m2GU3U=

YT3CqMsfbiuq+D3gG98MaVLeasNuoXriSSMnJQdgT3PJJ+teqeWg/
hFO6UDCsLxl4cj8W+EdR0SR9huY8I/8AdcEMp+m4Ct2igD5x0H4ha/
8ADKyHhrxP4duZUtCVgnjO3K5zgHG1xzwQfatb/hoTTf8AoXr/
AP7+LXu5APUZpPLT+6KBWPCf+GhNN/6F6/8A+/i0f8NCab/0L1//AN/Fr3by0/uijy0/
uigLHhP/AA0Jpg/5l+//aF0sHB8P33/
fxaj+NfjyXRfHGgWWn7WbSXW+nUNw7twEPp8mfwkq1qOs2/iTxp/
aVm262SJfIOOoxwfzYn8KAsQ/8NC6Yenh+/wD+/i0v/DQmm/8AQvX/
AP38WvVvC+kraWCySIPNk+Y+w7Cug8tP7P7ooCx4Cx4Nl+GbyW6YjV5BjP/
AAEEmtH4TeCta/tbUPF3iOJob3UUWLLC64ZVJ3EkdsnGB2Ar2jy0/
uinAAdKB2ADAxUN1bRXtXpNazruhmRo3X1UjBFTUUAfNum3fiX4X4
RIHoGDAEBiAMqfQY467X/DQmm/9C9f/wDfxa94IB603y0/uigVjwn/AIaE03/oXr//
AL+LR/w0Jpv/AEL1/wD9/Fr3by0/uijy0/uigLHhP/DQmm/9C9f/APfxaQ/tC6YOvh+/
H/bRa6T46+Kj4c8IQWNnKYtQ1GceWycFY4yGZs/XYMd9xrn9e8VW/
jVPDs8KqI3t1lniU5CyE/Ov4bSPxoCxH/w0Lpn/AEL9/wD9/Fpf+GhNN/6F6/8A+/
i16N4Q0verX86AvIflyPzNdh5af3RQFjwg/tB6eQnh2/Z+wMijJ/
Kq3hPQvEPxE+IMfjPW7B9M02yj2WFvICrPwQDggHA3MxbnE7fQHlp/
dFKAB0GKB2GxRiKJUHQCn0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUU
UUAFFFFAGJ4q8M2fizRxpmoSzrZeass0UTbzeass0UTbfOC8hGPXbnB4weByK8Y/
Z68NWl3BceIhLNFf2d00HyN8ksTRrlGX6nIIxyB16V9AP8A6tvoa8Z/Zu/5E/
Vv+wDaa0Ae0UUUUAFFFFABRRRQA1wzRsEbaxBAbGcH1xXzfqPw50GP44aDX/9eT/
ouagD1+0he3s4E+w+Tvy/ZmnL9Dj+e/zwVxhLGUB4sJ6BVUlQB29etavwt8Pxz7TnZuNsjnaGOcKD0/
EWd8KhwLpSa4GEXwLPd8ge8JZhP5B3yZXiXaiBxL1/XbEPCrlCf7wEQz2mdtpeMcrT+qvAVADZMZiQ/
AGh+BzJL7nWkcjGKaHOkDBOH0T3XUIvRtoHk4u7UIs3N84P7b8nz1+n+EwAMWs5GzeeH/
AKATOf8AoQxV32lPohg3OZm/2ceBC1EX+bm0iGdGxUSTrX10axWJWWEcmQrhF5Nt5yeAW12pl/
+WKSr3DjoRzgnbOm5u/Cjx/BP8AqBP8AQCI2WVHzgX7upJhA/UA/
mCB8dJLWWcG2tf4gOsmIKMVrGDVJWf6rdYlPyeJYiCn0lTdpMAaA/
qAkdTPrRuxxmqq4TIamtktNGlLyd7/O6/g6PxlLq0OpV+/vzvAM9m3C3pxmmfRMfdTGjcPZQa+zBVAHfp/
BYfxVuaVdHptcGMhymv+2RmBr/cEn/SfeT/Q3CS+aYmdtpKjpDApKg+YRM6adcbWh9GmHSsqKjv+Lx/
3ZiMsExAtPO/aoHQjPnzMM4ns/zdmEAoq0mBMP9W+tbjx0YDR51WJfI/EJEmrKsnL9aDrPzqAhx/
lN2AO3XIrr8qSDUXkPPt3WvIRvV5eD+SpuUg5ftMgsvE6Q6jQrBbUXNbvPpCVJmAHDxIYKZxYKXOX2Y/
7PwcV8vRhBlgMxjIs5+v1bZQm0ka8ZHIW2X/
ALtn6lAovE9PThQ5EXkB+ZOBNLP09UZZDaj6Iej1sQuZhEq16P9NFKCUpLY+xvr3sfr5N2bT4D8AR2Ms/
KFvjn8vxvhzcHFl6Y6ZcdF6GoiNC1h6sngnl5z4vTpr1hLBHWmqLdy7IHNqJDxOuW/v3tO/
ANbRyQj6+5rCzxJE+ZdNl0cb/ufRS4C/W6znl2/Nay5S4zvH+jSt/bbmSGX7MOD+zrp59gFybtxXrTbu
Fwzn2pnb5CcHsqqf/tpS0+BFWI2lXXx2/IbfZ+fVLrEeb7MG2v7D/ARtelUx5dA4LOKQoGt/cm/
Ck4j/ADK2Pd/QGyo/Zsg8xHCW3v+Es+m8dRkN+eVczvnbxD4GE9g7xj/X1mHAWB2UJRSGkBIRRETQg1v/
HCkjC+KMqvnvuJV4PSYaXVtmJjtmIm5lRAWEX2c4WeGHpvJiSDCv7ew5qpHc9/
AMlUUjY4QdtXrD0IM/lUBq8uiB28R9h5AWJr3O/VtBFl8SgvVEXS0dLW/
WVM0pzCG5cpHDOCKxo4gXBTYhGC1GuNBpRNFpAa9sXWRV1W4eTVtYdMTHsHJC6fX8v/
z1cWpwWeaB8HPnyJW8Su1zWHXLn7Wv/HnXEqMo14LDkUDK2oX2XwD/
KsyRXG8zc35NupC+mQ8FLzGHAwhT2XSOY8gLi7TBHhZYXdQCubl0JaOSK8R2ltMYoAHPyY1kIfcBzXYT/
h1eWaGXM/9bcC9OqmqyMenbT4Ycbb3dB+u4x5NjUQ8mgR3ZkD9Pfhc3S4Ltf5rAjYs3DJOxSsQKtq9ugN/
v8A+bPYqxTe0gx5Q4kVx1ZuF+mk+vG8x9JLeggdBmtH7eCsyEcwXbGiZHAkEXP3zXLIYzSZ+9tM5Wz/
WFDXN6v6uArWvNNRmKBxdHpv7Gso4K2Zxju8pU5O7dnDH2fKSXtbSGc1UHpIHKw5J9RM9ZsSknwpGCwX2BV/
3AAGRLzNF3l6NiLm0n9DWf5aFNZQXFD1t1sXcrbo7aEnL8+CGSDibXxMS8zaN/ErIRrGSe96ybzxo3/
JRrHNwX+DuaNAm8b1/3adKd05pQ+wgZoiUB4kZ2w+CSGd/P+aX0xI0TkSyqkTR4j4HYVi9C/
tpD6e3g+iX6Dx4+Kq3y9QfzpBhZyvaOKBm58iOwkIZ1vQfD4i5JnT7vlDIZwIa/0upfBkzhVm93qb/uJ/
i0nsaRMpZDzCSdzHZ1/wdq4D9UIbfLi+sq/E2+PWWN/SbhFR3N2qDhpRlGyCUhtQCCXJgjaFDWVs2YXXT/
oR1N+7TSqLkqGc2ULlqwrTkJqQVNEeu6WfFhCkRJjbfSJMw/oBzmN1avpqtGOOvcgzrHeTcLlsOoI3U1BrD/
dyPFjtZqnqz1yaOr+lBxkfHTuckixp/
Ej1ppNrS7jXNbldVgo6Uhq2v4X9a0mf5prxU3Sk2JNvuf4kOtv4I+FWZ2u5fzAyVBPzb7j/
W2BfffuUK2JOyFaPf0Xwrp/GSvGGXpBaHaaH+CIYq0XYcS8ZBrGCP3jcdiBWo62zbnDEV/
5nBWOr4j5JKCiOM2+K3m9vWfeLppbs/f7hb0nwd6Zz/AJ/RTVOSE2MdlhcjVa2AdyiZ9VO5N+yL/
EV5dkOwCYuPh2WMJs0mfQm7oQkl/gmRo0ubsLsUy5SQ3dXnnVhpp6pzpE/
+Rm3tzZI8M0fcR8Dy2smnTffBEKe6N2DmdX6TJXx/tFyjH/i88aleY1/FBZ2qKPgGyQWqPD/
Mn+LVwBkwEWMszytstzRdTXVXa8nBY1+IKrUmjfvrw/iQdSfp6WXzwtf3Jd6qbypkZ97fbOFRAEWzoevo/
Ua4FLTN5u4dwXsUVaSlW/y8mTWlswj0zsWMBffqAEtZNyUj3xv7cD1e2DfHn3m+C/
XlC5rvUJ9fn8wYufJgkT+/RG0/QG0A/IOGmA4T9ayRWxO/
ghNGrxQfbx2A8w0jAPlYCmtz0cB2OunoXGxf6KcytsYwIX6+fqz5+9RTNr7I41ZDF8yeXB9fE/
6R5drhM3KBn3WRQzrgd/HQiAhelfEXqGfdTMSO0U1vzyF+6WCeeqx48yfIZW1ebdPrb8XXdFHDJfX/
zAkP3Q5QrRdhdVnT0oDCa87ZbMakoZQzGMTh6VvRkPkTY+7pUkaN1g+fmXoPFjvxV+pJMfRgZwTjWZoF/
61n/ABWPeSfufOjc/XsNQ6f5w9Q1KU+KybPdKeP9Nt/BjLGKLoNp+Hl/6u6jRtaIgQJWcBu/ZNcSN/

QXrvvCKqvg3Q9oA/wCJfb9P+ua1wP7Q3/JNov8AsIRf+gvXf+E/+RN0P/
sH2/8A6LWgDYooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKA
CiiigAooooA43wnMlp4v8X6TMdl098moRq3HmQyQxqGX1AZGUkdDxXQ3uvaTp1xLBe6hb2
8sVubp0lfaRECQXGeoGOcdOM9RVXXvC1hr8kFxJJc2eoW2fs9/
ZS+XPED1APIKnurAj2rB1X4YWPiNETxFrmsatHErCGOaSKNYmIxvAjjXLDtuyPUGgC78NY
rhPAljLcwGB7qSe7WJjyqSzPIn/
jrCusqtp9vcWthDBdXZu5o12tOYwhf0JA4BxjOMD2HSrNABRRRQAUUUUAcZ8VFuH8A3K2r
olybuzETuMqr/AGmLBPtnFZl14Wv/AAdInirTLi71fU1X/
idLIcvqEWQSyJnarRgfIo7Dbznntdb0a317TDYXTypEZopsxEBsxyLIvUHjKjPtmtGgCpp
mp2es6Zbalp86z2lzGJIpF7g/
yPqDyDxVusjRvDtpoN1qMljLOsF7N55tGYGKFyPmMYxldx5IyRnpitegAooooAKKKKAM3x
Bo0HiHw9f6Rcge4I9xXBvqk/jLw94W8P3YBvb24/
wCJwjL91bRh54YDpulVF+j16dXP6X4X4N0rSPFGqeIbYTfbNRx5isw8uPpu2AAY3lVLZJyQK
AOgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooo
oAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA8U1j/AJOp0H/rxb/
0TNXtdeKax/ydToP/AF4t/wCiZq9roAKKKKACiiigAooooAKKKKACiiigDxr4y/8AI/
fDf/sIn/0bBXsteNfGX/kfvhv/AF4t/wCiZq9roAKKKKACiiigAooooAKKKKACiiigDxr4y/8AI/
wDo2CvZaACiiigAooooAKKKKACiiigAooooA8o/aG/5JtF/2EIv/QXrv/Cf/Im6H6H/
2D7f8A9FrXAftDf8AJNov+whF/wCgvXf+E/8AkTdD/wCwfb/
+i1oA2KKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA
KKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAC
iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAo
oooAKKKKACiiigAooooAKKKKACiiszWNctdEbT/
ALXHMUvbtLNJI1BWN3ztL88AkBc88kUAc3efD9rv4rWHjb+0gotSQ9JJhIP7d3c9lTHq4iJt
nau3orM0jXbXW5dRS0SbZYXTWjyuoCvIoG7ZzkgE7SSByD1oA06KKKACiiigAooooAKKKK
ACiiq2oXsem6bdX8yu0VtC8zhACxVQScZ78UAcp408BN4u8MZcI6x8ZFhtZCKYRxiHJdCTgH
YzuG3/
AFfv1rtKq6ZfxarpNqMCusN3Ak8auAGCsoYA4JGcH1qjp3iSz1fWb3T3T7CKeeOy+Se9RV+
zrL3iDZyzjqcAgdCQeKANiiiigAooooAKKKKACiiigAooooA5L4ieiezC488NLo634sitwk/
mmLzPuhhjGR/
e9a6HSLH+y9GsdPMpmFrbRweYRgvtULn+vsU9ba2cx3E1kkYhikHVD
JI6qWGRkKTjvVvRPFtjrV7VmzbW2KNpd5ZpmjmnNpZd+mlaEzi5NCKQfvB3z7rAwVeZ5LdhNm
UNDLBtbl/8rbVOgQ5pG/Ngi97HiSKqgW5TSYG1+dgTfTfpZDTMKkL2WZJ3Bk2ozALQ0sPJeowfwFHTtzxZ/
+KAN6iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACi
iigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoo
ooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKK
KACiiigAooooAKKKKACiiigArF8W6IfEfhTUtKR9k08J8h8kbJV+aNsjkYcKfwraooA5JP
Gka/
DFvFckYEsVmzyQMCuLhfkMXrnzAVrS8IaM+geFrGwncyXSoZbqQnO+dyXkOfd2b8K4t9D1
UfEBvD62U/
wDwjcmorrxuQG8vO0loS3TP2gLJs9CTXp9ABRRRQAUUUUAFFFFABRRRQAVkeK/+RO1v/
sHz/wDotq16y/E0Uk/
hTWIYY3klkksZlREUlmYoQAAOpoA8qi8XXF94N8NWkC3lj4VVWG94YIgFw7Tyrob8I

```
0uMAkgHvXsGn2FnpenwWVhBHBaQoFijjGFUf571leHNNR/AOkaZqFp8jaZDBcW0yY/
5ZAMrKfxBFZnhSPVPDup3Hhe9jubrTYl83StQKs4EOeYJW5wydFJPzLjpjFAHY0UUUAFFF
FABRRRQAUUUUAFYfjS/n0vwRrt9asUuILCaSJx/
CwQ4P4Hmtyq2o2MGqaZdafdKWt7qF4ZVBxlWBB/
Q0AVfD2k22heHdO0u0AEFrbpGpxjdgcsfcnJPuawfHsQt5fDusQER3tpq9vCsgHLRTOIpE
+hDZ+qg9qqaFr2o+FdPg0LxLpmpTSWaCKDUrG0kuYrqJeFY+WGZHxwVYdRkE5rn7PUNc8Q
+K9Gt/ElndRaBaXrXFjez2TQPeXCgCESp/
wAs8FnIJ2hyowO1AHrlFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFF
ABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQ
AUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUA
FFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAB
RRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU
UUUAFFFFAH//Z',
  'image_mime_type': 'image/jpeg'}}
```


Separate extracted elements into tables, text, and images

```python
# separate tables from texts
tables = []
texts = []

for chunk in chunks:
    if "Table" in str(type(chunk)):
        tables.append(chunk)

    if "CompositeElement" in str(type((chunk))):
        texts.append(chunk)

# Get the images from the CompositeElement objects
def get_images_base64(chunks):
    images_b64 = []
    for chunk in chunks:
        if "CompositeElement" in str(type(chunk)):
            chunk_els = chunk.metadata.orig_elements
            for el in chunk_els:
                if "Image" in str(type(el)):
                    images_b64.append(el.metadata.image_base64)
    return images_b64

images = get_images_base64(chunks)
```

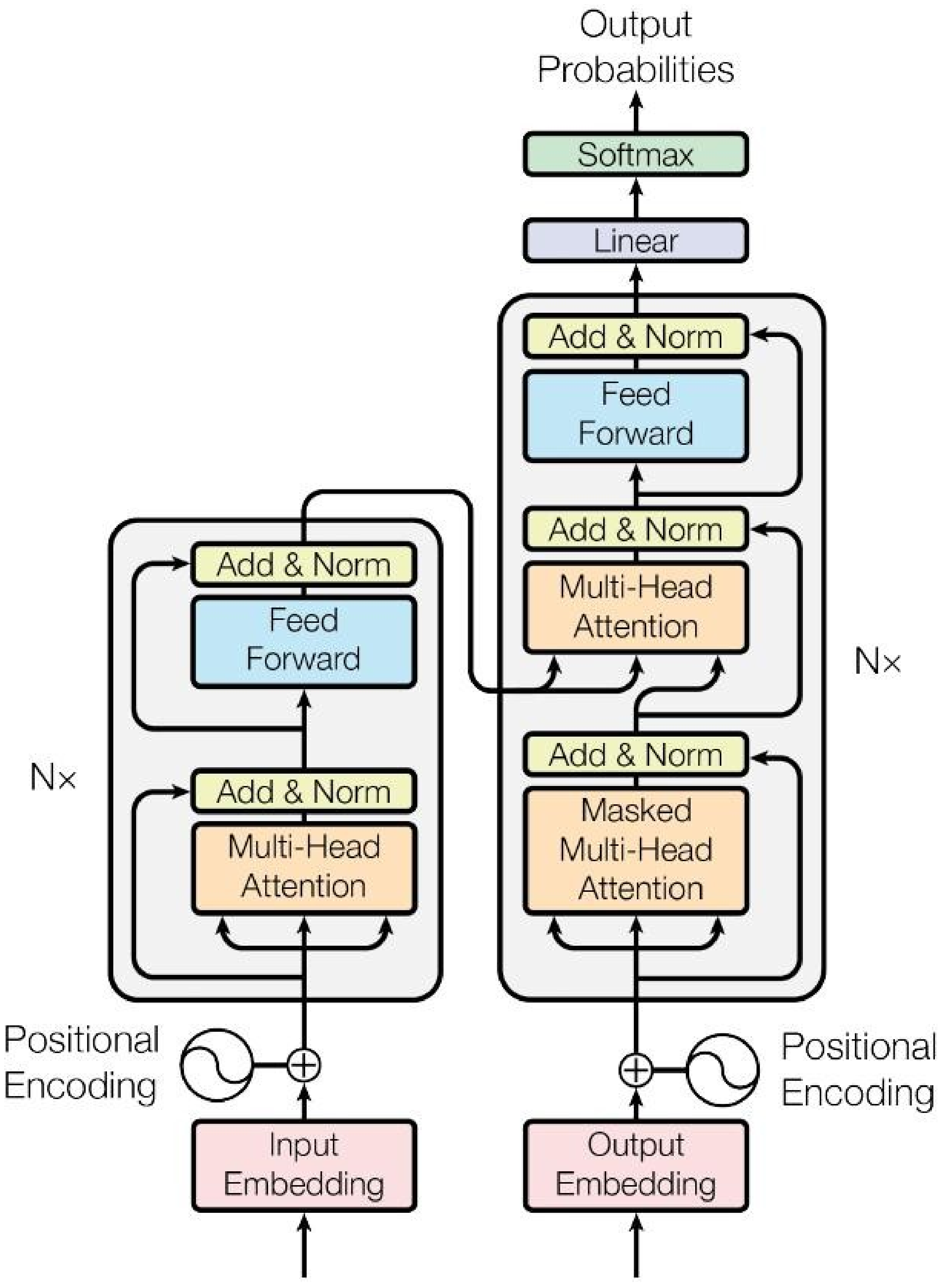
Check what the images look like

```python
import base64
from IPython.display import Image, display

def display_base64_image(base64_code):
    # Decode the base64 string to binary
    image_data = base64.b64decode(base64_code)
    # Display the image
```

```
    display(Image(data=image_data))

display_base64_image(images[0])
```


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Output
Embedding
Positional
Encoding

Summarize the data

Create a summary of each element extracted from the PDF. This summary will be vectorized and used in the retrieval process.

Text and Table summaries

We don't need a multimodal model to generate the summaries of the tables and the text. I will use open source models available on Groq.

```
%pip install -Uq langchain-groq
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
from langchain_groq import ChatGroq
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser

# Prompt
prompt_text = """
You are an assistant tasked with summarizing tables and text.
Give a concise summary of the table or text.

Respond only with the summary, no additionnal comment.
Do not start your message by saying "Here is a summary" or anything
like that.
Just give the summary as it is.

Table or text chunk: {element}

"""
prompt = ChatPromptTemplate.from_template(prompt_text)

# Summary chain
model = ChatGroq(temperature=0.5, model="llama-3.1-8b-instant")
summarize_chain = {"element": lambda x: x} | prompt | model |
StrOutputParser()

# Summarize text
text_summaries = summarize_chain.batch(texts, {"max_concurrency": 3})

# Summarize tables
tables_html = [table.metadata.text_as_html for table in tables]
table_summaries = summarize_chain.batch(tables_html,
{"max_concurrency": 3})
```

```
text_summaries
```

```
['The paper "Attention Is All You Need" by Ashish Vaswani et al.
introduces the Transformer model, a new network architecture based on
attention mechanisms, replacing traditional encoder-decoder models.
```

```
The Transformer achieves state-of-the-art results on two machine
translation tasks, improving over existing best results, and
generalizes well to other tasks, such as English constituency
parsing.',
 'Recurrent neural networks have been established as state of the art
in sequence modeling and transduction problems, but their sequential
nature limits parallelization and requires computational efficiency
improvements through factorization tricks and conditional computation.
Attention mechanisms have improved sequence modeling, but typically
require a recurrent network. The Transformer model proposes to use
attention mechanisms entirely, eschewing recurrence and allowing for
more parallelization and improved translation quality.',
 'The Transformer model has an encoder-decoder structure, where the
encoder maps input symbols to continuous representations and the
decoder generates output symbols one element at a time. The model
architecture consists of stacked self-attention and point-wise, fully
connected layers for both the encoder and decoder. The encoder and
decoder stacks each have 6 identical layers with two or three sub-
layers, using residual connections and layer normalization.',
 'The attention function maps a query and key-value pairs to an
output, computed as a weighted sum of the values, where the weights
are the compatibility function of the query with the keys. There are
two main types: Scaled Dot-Product Attention and Additive Attention.
The Scaled Dot-Product Attention uses a dot product of the query with
the keys, scaled by √dk, and applies a softmax function to obtain the
weights. This method is faster and more space-efficient than Additive
Attention, but Additive Attention outperforms it for larger key
dimensions.',
 'Multi-head attention is a method that linearly projects queries,
keys and values h times with different learned linear projections,
then performs attention in parallel, allowing the model to jointly
attend to information from different representation subspaces at
different positions. The output values are concatenated and projected
to yield final values. The Transformer model uses multi-head attention
in encoder-decoder attention layers, self-attention layers in the
encoder, and self-attention layers in the decoder, with a total of 8
parallel attention layers.',
 'The encoder and decoder layers in the model contain a fully
connected feed-forward network (FFN) with two linear transformations
and a ReLU activation in between. The FFN is applied separately to
each position, with different parameters for each layer. The model
uses learned embeddings to convert input and output tokens to vectors
and applies a linear transformation and softmax function to predict
next-token probabilities. The embedding layers share the same weight
matrix with the pre-softmax linear transformation.',
 'Positional encoding is used to incorporate sequence order into a
model, as traditional models lack recurrence or convolution. A
sinusoidal function is used to create positional encoding, which is
hypothesized to allow the model to learn to attend by relative
```

```
positions. The model can easily learn to attend by relative positions
since the sinusoidal function of the position can be represented as a
linear function of the sinusoidal function of the position offset.
This allows the model to handle longer sequence lengths.',
 'The training regime for our models involved using standard WMT
datasets, byte-pair encoding, and batching by approximate sequence
length. Models were trained on 8 NVIDIA P100 GPUs with the Adam
optimizer and varied learning rates. Training times ranged from 12
hours for base models to 3.5 days for big models.',
 'The model uses residual dropout and label smoothing to improve
accuracy and BLEU score. The big transformer model achieves a new
state-of-the-art BLEU score of 28.4 on English-to-German and 41.0 on
English-to-French translation tasks, outperforming previous models at
a lower training cost.',
 "The Transformer model's importance was evaluated by varying its
components and measuring performance changes on English-to-German
translation. The base model was modified in different ways, with some
components having varying TFLOPS values, and results were measured on
a development set.",
 'The study investigates the performance of the Transformer model in
different settings. It varies the number of attention heads, attention
key and value dimensions, and model size to find the best settings for
the model. The results show that single-head attention is worse than
the best setting, reducing the attention key size hurts model quality,
and bigger models with dropout are better. The model also generalizes
well to English constituency parsing, achieving good results in both
supervised and semi-supervised settings.',
 'The Transformer model, the first sequence transduction model based
entirely on attention, outperforms previous models in translation
tasks, achieving a new state of the art in English-to-German and
English-to-French translation tasks.',
 "Research papers on computer vision and neural machine translation
are cited, including works by Szegedy et al. on the inception
architecture, Vinyals et al. on grammar as a foreign language, Wu et
al. on Google's neural machine translation system, and Zhu et al. on
shift-reduce constituent parsing. Figures 3, 4, and 5 illustrate the
attention mechanism in neural networks, showing attention heads
attending to distant dependencies and exhibiting behavior related to
sentence structure."]
```

Image summaries

We will use gpt-4o-mini to produce the image summaries.

```
%pip install -Uq langchain_openai

Note: you may need to restart the kernel to use updated packages.
```

```python
from langchain_openai import ChatOpenAI

prompt_template = """Describe the image in detail. For context,
                  the image is part of a research paper explaining the
transformers
                  architecture. Be specific about graphs, such as bar
plots."""
messages = [
    (
        "user",
        [
            {"type": "text", "text": prompt_template},
            {
                "type": "image_url",
                "image_url": {"url": "data:image/jpeg;base64,
{image}"},
            },
        ],
    )
]

prompt = ChatPromptTemplate.from_messages(messages)

chain = prompt | ChatOpenAI(model="gpt-4o-mini") | StrOutputParser()


image_summaries = chain.batch(images)

image_summaries
```

```
['The image presents a schematic representation of the transformer
architecture, which is fundamental in natural language processing and
machine learning. Here’s a detailed description of its components:\n\
n1. **Overall Structure**: The diagram is divided into two main
sections, representing the encoder (on the left) and the decoder (on
the right) of the transformer model. Both sections involve repeated
layers denoted as \\(N_x\\), indicating that these layers are stacked
multiple times.\n\n2. **Positional Encoding**: At the bottom of both
sections, there are blocks labeled "Positional Encoding." This
component is crucial as it adds information about the position of
tokens in the sequence, enabling the model to understand the order of
inputs.\n\n3. **Input and Output Embeddings**: Below the positional
encoding blocks, there are two labeled boxes: “Input Embedding” for
the encoder and “Output Embedding” for the decoder. These embeddings
transform the input tokens into continuous vector representations.\n\
n4. **Encoder Side**:\n   - **Multi-Head Attention**: The first major
block labeled "Multi-Head Attention" is responsible for capturing
relationships between different tokens in the input sequence. It is
connected to an "Add & Norm" block, which combines its output with the
original input and normalizes it.\n   - **Feed Forward**: Following
```

```
this is the "Feed Forward" block, where a fully connected neural
network processes the data further. This is also followed by another
"Add & Norm" layer, ensuring that the outputs are again combined with
the original inputs.\n   - This structure is repeated \\(N_x\\) times
to create multiple layers.\n\n5. **Decoder Side**:\n   - **Masked
Multi-Head Attention**: Similar to the encoder, the decoder starts
with a "Masked Multi-Head Attention" block. The masking is crucial to
ensure that the predictions for a particular position can depend only
on the known outputs before it.\n   - **Feed Forward**: It also
includes a "Feed Forward" block followed by an "Add & Norm" block.\n
- The decoder then connects to the output probabilities through a
"Linear" layer followed by a "Softmax" layer, which converts the final
output into probabilities for each token in the vocabulary.\n\n6.
**Arrows and Flow**: The diagram uses arrows to indicate the flow of
data throughout the architecture, showing how inputs are transformed
and passed between layers.\n\nOverall, the schematic effectively
illustrates the flow of information through the transformer
architecture, highlighting the importance of attention mechanisms and
normalization in processing sequences.',
 'The image depicts two main concepts from the transformer
architecture: **Scaled Dot-Product Attention** and **Multi-Head
Attention**.\n\n### Scaled Dot-Product Attention (Left Side)\n1.
**Structure**: \n   - The flow starts with three inputs represented as
**Q** (Query), **K** (Key), and **V** (Value).\n   - The first
operation is a **MatMul** (matrix multiplication) between Q and K,
producing a score.\n   - This score is then passed to a **Scale**
component, which adjusts the magnitude of the scores.\n   - An
optional **Mask** is applied next to prevent certain positions from
being attended to, often used in tasks like language modeling.\n   -
The scores are then passed through a **SoftMax** layer to convert them
into probabilities.\n   - Finally, another **MatMul** operation takes
the output from SoftMax and multiplies it by V to produce the final
attention output.\n\n### Multi-Head Attention (Right Side)\n1.
**Structure**:\n   - This section builds upon the scaled dot-product
attention.\n   - It begins with three linear transformations applied
to V, K, and Q, indicated by three **Linear** blocks.\n   - These
transformed vectors are then processed by the **Scaled Dot-Product
Attention** block (highlighted in purple).\n   - The outputs from
multiple attention heads are concatenated using the **Concat**
operation.\n   - The final output from this multi-head attention
structure is connected to another **Linear** layer.\n\n### Visual
Elements:\n- **Color Coding**: Different components are color-coded
for clarity, with purple for the main attention mechanism and other
colors for supporting operations.\n- **Arrows**: Directional arrows
indicate the flow of data through the various operations,
demonstrating how inputs are transformed at each stage.\n- **Labels**:
Clear labels for each component help in understanding the
functionality of each part of the architecture.\n\nThis diagram
effectively illustrates how attention mechanisms are implemented in
```

```
transformers, showcasing the sequential and parallel processing
involved.',
 'The image depicts a diagram illustrating the architecture of the
transformer model, specifically focusing on the scaled dot-product
attention mechanism.\n\n### Components:\n\n1. **Input Vectors (V, K,
Q)**:\n   - At the bottom of the diagram, there are three labeled
boxes: **V** (Values), **K** (Keys), and **Q** (Queries). Each box is
associated with a "Linear" label above it, indicating that these
inputs undergo a linear transformation.\n\n2. **Linear
Transformations**:\n   - Above each of the V, K, and Q boxes, there
are "Linear" labels. This signifies that each input vector is
processed through a linear layer, typically involving a weight matrix
multiplication and an optional bias.\n\n3. **Scaled Dot-Product
Attention**:\n   - Central to the diagram is a larger box labeled
**Scaled Dot-Product Attention**, highlighted in purple. This is the
core operation that computes attention scores based on the queries,
keys, and values.\n\n4. **Concatenation**:\n   - Above the attention
box, there is a "Concat" box, suggesting that the outputs from the
attention mechanism may be concatenated with other features or outputs
in the subsequent layers.\n\n5. **Flow Arrows**:\n   - Several arrows
indicate the flow of data:\n     - Arrows point upwards from the
linear transformations (V, K, Q) to the scaled dot-product attention
box, showing that these inputs are fed into the attention mechanism.\n
- There are also arrows pointing from the attention box to the
Concatenation box, indicating that the output of the attention
mechanism is directed toward concatenation.\n\n6. **Output (h)**:\n
- An arrow extends from the attention box to a label **h**, which
likely represents the output of the attention mechanism or the hidden
state that will be used in further processing.\n\n### Summary:\nThe
diagram effectively illustrates the key components and flow of
information in the scaled dot-product attention mechanism of the
transformer architecture, highlighting the importance of linear
transformations and the interaction between inputs and the attention
mechanism.',
 'The image appears to illustrate a component of the transformer
architecture, likely focusing on attention mechanisms. It features a
series of words arranged vertically and horizontally, representing
tokens in a sequence. \n\n1. **Token Representation**: The words are
aligned in rows and columns, with some words repeated, suggesting the
model processes multiple input sequences. Key tokens like "making,"
"the," "registration," and "more difficult" stand out, possibly
indicating significant elements in the context.\n\n2. **Attention
Weights**: There are colored lines connecting specific words,
indicating the attention weights assigned during processing. The lines
vary in thickness and color, suggesting different levels of attention
or importance that the model attributes to these tokens in relation to
others. For instance, thicker lines might denote stronger
relationships, while thinner ones indicate weaker connections.\n\n3.
**Special Tokens**: The sequence includes special tokens such as
```

```
`<EOS>` (end of sequence) and `<pad>` (padding), which are common in
transformer architectures for managing input lengths and denoting
sequence boundaries.\n\n4. **Layout and Design**: The layout is
organized and structured, with a clear focus on how different tokens
interact. The use of color helps distinguish between different types
of connections or relationships among the words.\n\nOverall, the image
serves to visually represent how the transformer model attends to
different parts of an input sequence, emphasizing the relationships
between specific tokens as part of the model\'s processing
mechanism.',
 'The image depicts a visual representation of the attention mechanism
commonly used in transformer architectures. It features a series of
words arranged in a horizontal layout, each accompanied by a
connecting line that illustrates the attention weights between them.\
n\n### Key Features:\n\n1. **Words/Tokens**: The image contains a
sequence of words, which seem to form part of a sentence. These words
are positioned in a linear fashion, creating a clear progression from
left to right.\n\n2. **Attention Lines**: \n   - **Lines and Colors**:
Various lines connect the words, indicating how attention is
distributed across different tokens. The lines are primarily in shades
of purple, with varying opacities.\n   - **Thicker Lines**: Some lines
are thicker, suggesting stronger attention weights between specific
tokens. For example, connections from the word "Law" to multiple other
words indicate its importance in the context.\n   -
**Directionality**: The lines originate from words on the left and
connect to those on the right, representing how each word influences
others in the sequence.\n\n3. **Special Tokens**: \n   - There are
tokens such as `<EOS>` (End of Sequence) and `<pad>` (padding), which
indicate the beginning and end of the processing sequence. These are
positioned at the far right, contributing to the overall structure.\n\
n4. **Layout**: The layout appears somewhat tree-like, with several
branches stemming from particular words, showing how certain words
(like "Law" and "application") interact with multiple other tokens.\n\
nThis visual effectively conveys how the transformer architecture
utilizes attention mechanisms to weigh the relevance of different
tokens in a sequence, which is crucial for understanding contextual
relationships in language processing.',
 'The image illustrates a visualization related to the transformer
architecture, likely depicting attention mechanisms. It features
several words arranged in two rows, with connections represented by
lines between them.\n\n### Description:\n\n1. **Words**:\n   - The
left side contains a sequence of words: "The," "Law," "will," "never,"
"be," "perfect," "but," "its," "application," "should," "be," "just."\
n   - The right side includes: "this," "is," "what," "we," "are,"
"missing," "in," "my," "opinion," along with special tokens "<EOS>"
and "<pad>".\n\n2. **Connections**:\n   - There are multiple green
lines connecting the words from the left and right sides. The lines
vary in thickness, indicating the strength of attention or relevance
between the words. \n   - Thicker lines likely represent stronger
```

```
connections, suggesting that those word pairs have a higher attention
score.\n\n3. **Color and Transparency**:\n   - The lines are in
different shades of green, with some being more opaque and others
lighter, which may indicate varying levels of significance in the
attention mechanism.\n\n4. **Layout**:\n   - The words are arranged in
a horizontal layout, with connections crisscrossing between the two
groups, visually representing how the transformer model attends to
different parts of the input sequence.\n\nThis visualization
effectively showcases how a transformer model processes and relates
different words within a sequence, emphasizing the attention
mechanism\'s role in understanding context and relationships within
language data.',
 'The image appears to illustrate concepts related to the
transformer\'s attention mechanism, commonly used in natural language
processing. \n\n### Key Features of the Image:\n\n1. **Text
Representation**: \n   - The image consists of a series of words
arranged in a linear sequence, representing a sentence or phrase. The
words include terms like "The," "Law," "will," "never," and so on,
extending to phrases like "in my opinion" and special tokens such as
`<EOS>` (end of sequence) and `<pad>` (padding).\n\n2.
**Connections**: \n   - There are numerous lines connecting the words
across the two sections of text. These lines vary in thickness and
color intensity, indicating the strength of attention between the
words.\n   - The thicker, darker red lines suggest a stronger
relationship or attention between specific words, while the lighter
lines indicate weaker relationships.\n\n3. **Attention Mechanism**:\n
- The lines likely represent how the transformer\'s attention
mechanism works, showing which words pay attention to others during
processing. For example, a word on the left may direct its attention
toward multiple words on the right, indicating contextual
relationships or dependencies.\n\n4. **Layout**:\n   - The words are
organized in a two-row format, with the first row containing the
initial part of the sequence and the second row representing
subsequent words. This layout emphasizes the connections across
different parts of the sentence.\n\n5. **Implications**:\n   - This
visual representation is intended to help readers understand how
transformers manage relationships between words through attention,
which is crucial for tasks such as translation, summarization, or
question answering.\n\nOverall, the image serves as a visual aid to
explain the complex interactions among words in a sentence as
understood by transformer models.']
```

```python
print(image_summaries[1])
```

```
The image depicts two main concepts from the transformer architecture:
**Scaled Dot-Product Attention** and **Multi-Head Attention**.

### Scaled Dot-Product Attention (Left Side)
1. **Structure**:
   - The flow starts with three inputs represented as **Q** (Query),
```

```
**K** (Key), and **V** (Value).
   - The first operation is a **MatMul** (matrix multiplication)
between Q and K, producing a score.
   - This score is then passed to a **Scale** component, which adjusts
the magnitude of the scores.
   - An optional **Mask** is applied next to prevent certain positions
from being attended to, often used in tasks like language modeling.
   - The scores are then passed through a **SoftMax** layer to convert
them into probabilities.
   - Finally, another **MatMul** operation takes the output from
SoftMax and multiplies it by V to produce the final attention output.

### Multi-Head Attention (Right Side)
1. **Structure**:
   - This section builds upon the scaled dot-product attention.
   - It begins with three linear transformations applied to V, K, and
Q, indicated by three **Linear** blocks.
   - These transformed vectors are then processed by the **Scaled Dot-
Product Attention** block (highlighted in purple).
   - The outputs from multiple attention heads are concatenated using
the **Concat** operation.
   - The final output from this multi-head attention structure is
connected to another **Linear** layer.

### Visual Elements:
- **Color Coding**: Different components are color-coded for clarity,
with purple for the main attention mechanism and other colors for
supporting operations.
- **Arrows**: Directional arrows indicate the flow of data through the
various operations, demonstrating how inputs are transformed at each
stage.
- **Labels**: Clear labels for each component help in understanding
the functionality of each part of the architecture.

This diagram effectively illustrates how attention mechanisms are
implemented in transformers, showcasing the sequential and parallel
processing involved.
```

Load data and summaries to vectorstore

Create the vectorstore

```python
import uuid
from langchain.vectorstores import Chroma
from langchain.storage import InMemoryStore
from langchain.schema.document import Document
from langchain.embeddings import OpenAIEmbeddings
from langchain.retrievers.multi_vector import MultiVectorRetriever
```

```
# The vectorstore to use to index the child chunks
vectorstore = Chroma(collection_name="multi_modal_rag",
embedding_function=OpenAIEmbeddings())

# The storage layer for the parent documents
store = InMemoryStore()
id_key = "doc_id"

# The retriever (empty to start)
retriever = MultiVectorRetriever(
    vectorstore=vectorstore,
    docstore=store,
    id_key=id_key,
)
```

```
/var/folders/lh/7g2mv_x16p79z2rd9jqxqx_w0000gn/T/
ipykernel_92745/278287695.py:9: LangChainDeprecationWarning: The class
`OpenAIEmbeddings` was deprecated in LangChain 0.0.9 and will be
removed in 1.0. An updated version of the class exists in
the :class:`~langchain-openai package and should be used instead. To
use it run `pip install -U :class:`~langchain-openai` and import as
`from :class:`~langchain_openai import OpenAIEmbeddings``.
  vectorstore = Chroma(collection_name="multi_modal_rag",
embedding_function=OpenAIEmbeddings())
/var/folders/lh/7g2mv_x16p79z2rd9jqxqx_w0000gn/T/ipykernel_92745/27828
7695.py:9: LangChainDeprecationWarning: The class `Chroma` was
deprecated in LangChain 0.2.9 and will be removed in 1.0. An updated
version of the class exists in the :class:`~langchain-chroma package
and should be used instead. To use it run `pip install -
U :class:`~langchain-chroma` and import as
`from :class:`~langchain_chroma import Chroma``.
  vectorstore = Chroma(collection_name="multi_modal_rag",
embedding_function=OpenAIEmbeddings())
```

Load the summaries and link the to the original data

```
# Add texts
doc_ids = [str(uuid.uuid4()) for _ in texts]
summary_texts = [
    Document(page_content=summary, metadata={id_key: doc_ids[i]}) for
i, summary in enumerate(text_summaries)
]
retriever.vectorstore.add_documents(summary_texts)
retriever.docstore.mset(list(zip(doc_ids, texts)))

# Add tables
table_ids = [str(uuid.uuid4()) for _ in tables]
summary_tables = [
    Document(page_content=summary, metadata={id_key: table_ids[i]})
for i, summary in enumerate(table_summaries)
```

```
]
retriever.vectorstore.add_documents(summary_tables)
retriever.docstore.mset(list(zip(table_ids, tables)))

# Add image summaries
img_ids = [str(uuid.uuid4()) for _ in images]
summary_img = [
    Document(page_content=summary, metadata={id_key: img_ids[i]}) for
i, summary in enumerate(image_summaries)
]
retriever.vectorstore.add_documents(summary_img)
retriever.docstore.mset(list(zip(img_ids, images)))
```

Check retrieval

```
# Retrieve
docs = retriever.invoke(
    "who are the authors of the paper?"
)

for doc in docs:
    print(str(doc) + "\n\n" + "-" * 80)
```

```
[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon
Shlens, and Zbigniew Wojna. Rethinking the inception architecture for
computer vision. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as
a foreign language. In Advances in Neural Information Processing
Systems, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad
Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus
Macherey, et al. Google's neural machine translation system: Bridging
the gap between human and machine translation. arXiv preprint
arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep
recurrent models with fast-forward connections for neural machine
translation. CoRR, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu.
Fast and accurate shift-reduce constituent parsing. In Proceedings of
the 51st Annual Meeting of the ACL (Volume 1: Long Papers), pages 434–
443. ACL, August 2013.

12

Attention Visualizations
```

2 i i= RE 3 2 i 2 = = 2c 3 2 £ om % S GBANAAAA FS fe} oD DOD *H o Poe Q €oe2s oyzveyzuyeys 2SE @T_ EFESeSsEzESHR PL, SSSe TZSsSsggsg S=@LEGDEwB ECC oF aAeC RGN ESLSSSEESC -.VvVVVV VV HMO KEBOCSRSHLHOD QKXRDXE EO “A AAAAAA “= <2 £ 8 FogesouggsS ss P25 5273 Qvryxapvs\3 es sa 5 Seeneteecorzgrs 2g ogs aaa oO 2 Sere =~ aA o ° 8 ueeeogoa0 o © £ 5 ane) vvvvv Vv £ eg € ° 2 Ss v Do <¢ 8 & |

Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

13

<ped> <ped> UOIUIdO == Aw ul Bulssiw ale « aM = yeum = S| sy ysnf pinoys = uoluldo Aw ul Bulssiw ae ysnf 38q Pinoys uojeojdde si! nq poped 38q JaAou Me] au <ped> <SOa> uojuido Aw ul Bulssiuw oe aM yeum S| SIU} ysnf 3q Pinoys uojeodde Ss}! ynq yoped 3q 4eAeuU meq auL <ped> <SOa> uo|uldo Aw ul Bulssiuw oe eM yeum S| Siu} ysnf 3q Pinoys uoyeoydde si! ynq yoped 3q aul

Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

14

<ped> <ped> <SOH>\ <SO3> uoluido = uoluido Aw Aw yeyum S| sim pi—f}— 4 -ysn{ | Pinoys «+ pinoys uoeoydde uojeodde SHI S$}! | ya | nga ynq poped pooped aq aq Janou™ J@AoU WIM TIM me) me) aul “OU

<ped> <ped> so <0 Uo|UIdO uoluido Aw Aw ul ul Bulssiw Bulssiw ae ale aM am yeuM yeum S| S| sty} # sly -—A - el eq eq pinoys « pinoys uojeoidde ee Ss}! Ss}! nq ee popod a —_ ee eq eq JOAoU JOAoU IW IW rr auL auL

Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

15

Parser Training WSJ 23 F1 Vinyals & Kaiser el al. (2014) [37] WSJ only, discriminative 88.3 Petrov et al. (2006) [29] WSJ only, discriminative 90.4 Zhu et al. (2013) [40] WSJ only, discriminative 90.4 Dyer et al. (2016) [8] WSJ only, discriminative 91.7 Transformer (4 layers) WSJ only, discriminative 91.3 Zhu et al. (2013) [40] semi-supervised 91.3 Huang & Harper (2009) [14] semi-supervised 91.3 McClosky et al. (2006) [26] semi-supervised 92.1 Vinyals & Kaiser el al. (2014) [37] semi-supervised 92.1 Transformer (4 layers) semi-supervised 92.7 Luong et al. (2015) [23] multi-task 93.0 Dyer et al. (2016) [8] generative 93.3

3

2023

2

0

2

g u A 2] L C . s c [7 v 2 6 7 3 0 . 6 0

7

1

:

v

arXiv

i

X

r

a

Attention Is All You Need

Ashish Vaswani∗

Google Brain

avaswani@google.com

Noam Shazeer∗ Google Brain noam@google.com

Niki Parmar∗ Google Research nikip@google.com

Jakob Uszkoreit∗

Google Research usz@google.com

Llion Jones∗

Google Research llion@google.com

Aidan N. Gomez∗ † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser∗ Google Brain lukaszkaiser@google.com

Illia Polosukhin∗ ‡

illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English- to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

∗Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and

```
tensor2tensor. Llion also experimented with novel model variants, was
responsible for our initial codebase, and efficient inference and
visualizations. Lukasz and Aidan spent countless long days designing
various parts of and implementing tensor2tensor, replacing our earlier
codebase, greatly improving results and massively accelerating our
research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017),
Long Beach, CA, USA.

-----------------------------------------------------------------------
----------
N dmodel dff h dk dv Pdrop ϵls train steps PPL (dev) BLEU params (dev)
×106 base 6 512 2048 8 64 64 0.1 0.1 100K 4.92 25.8 65 1 512 512 5.29
24.9 (A) 4 16 128 32 128 32 5.00 4.91 25.5 25.8 32 16 16 5.01 25.4 (B)
16 32 5.16 5.01 25.1 25.4 58 60 2 6.11 23.7 36 4 5.19 25.3 50 8 4.88
25.5 80 (C) 256 32 32 5.75 24.5 28 1024 128 128 4.66 26.0 168 1024
5.12 25.4 53 4096 4.75 26.2 90 0.0 5.77 24.6 (D) 0.2 0.0 4.95 4.67
25.5 25.3 0.2 5.47 25.7 (E) positional embedding instead of sinusoids
4.92 25.7

-----------------------------------------------------------------------
----------
```

RAG pipeline

```python
from langchain_core.runnables import RunnablePassthrough,
RunnableLambda
from langchain_core.messages import SystemMessage, HumanMessage
from langchain_openai import ChatOpenAI
from base64 import b64decode


def parse_docs(docs):
    """Split base64-encoded images and texts"""
    b64 = []
    text = []
    for doc in docs:
        try:
            b64decode(doc)
            b64.append(doc)
        except Exception as e:
            text.append(doc)
    return {"images": b64, "texts": text}
```

```python
def build_prompt(kwargs):

    docs_by_type = kwargs["context"]
    user_question = kwargs["question"]

    context_text = ""
    if len(docs_by_type["texts"]) > 0:
        for text_element in docs_by_type["texts"]:
            context_text += text_element.text

    # construct prompt with context (including images)
    prompt_template = f"""
    Answer the question based only on the following context, which can
include text, tables, and the below image.
    Context: {context_text}
    Question: {user_question}
    """

    prompt_content = [{"type": "text", "text": prompt_template}]

    if len(docs_by_type["images"]) > 0:
        for image in docs_by_type["images"]:
            prompt_content.append(
                {
                    "type": "image_url",
                    "image_url": {"url": f"data:image/jpeg;base64,
{image}"},
                }
            )

    return ChatPromptTemplate.from_messages(
        [
            HumanMessage(content=prompt_content),
        ]
    )


chain = (
    {
        "context": retriever | RunnableLambda(parse_docs),
        "question": RunnablePassthrough(),
    }
    | RunnableLambda(build_prompt)
    | ChatOpenAI(model="gpt-4o-mini")
    | StrOutputParser()
)

chain_with_sources = {
    "context": retriever | RunnableLambda(parse_docs),
    "question": RunnablePassthrough(),
```

```
} | RunnablePassthrough().assign(
    response=(
        RunnableLambda(build_prompt)
        | ChatOpenAI(model="gpt-4o-mini")
        | StrOutputParser()
    )
)

response = chain.invoke(
    "What is the attention mechanism?"
)

print(response)
```

```
The attention mechanism is a function that maps a query and a set of
key-value pairs to an output, where the query, keys, values, and
output are all vectors. The output is computed as a weighted sum of
the values, with the weights determined by a compatibility function
that measures how well the query aligns with each key.

In mathematical terms, the attention function can be expressed as:

\[ \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\
sqrt{d_k}}\right)V \]

Here, \(Q\) represents the queries, \(K\) the keys, and \(V\) the
values. The dot products of the queries and keys are scaled by the
square root of the dimension of the keys (\(d_k\)) to prevent large
values that could push the softmax function into regions with small
gradients. This attention mechanism allows the model to focus on
different parts of the input sequence when producing an output,
enhancing its ability to handle dependencies and context within the
data.

Additionally, multi-head attention extends this concept by performing
multiple attention operations in parallel, allowing the model to
attend to information from different representation subspaces
simultaneously.
```

```
response = chain_with_sources.invoke(
    "What is multihead?"
)

print("Response:", response['response'])

print("\n\nContext:")
for text in response['context']['texts']:
    print(text.text)
    print("Page number: ", text.metadata.page_number)
    print("\n" + "-"*50 + "\n")
```

```python
for image in response['context']['images']:
    display_base64_image(image)
```

```
Response: Multi-head attention is a mechanism used in the Transformer
model that allows the model to focus on different parts of the input
sequence simultaneously. Instead of using a single attention function,
multi-head attention uses multiple attention heads, each performing
attention with different, learned linear projections of the queries,
keys, and values.

Here's a breakdown of how it works:

1. **Linear Projections**: The queries (Q), keys (K), and values (V)
are projected into different subspaces using learned linear
transformations. Each head has its own set of projections.

2. **Parallel Attention**: Each head computes attention independently,
allowing the model to attend to various representation subspaces from
different positions in the input sequence.

3. **Concatenation**: The outputs from all heads are concatenated
together.

4. **Final Linear Projection**: The concatenated output is then
projected again to produce the final output.

This approach enables the model to capture a richer set of
relationships in the data compared to a single attention head, which
might average out important information. Multi-head attention is
particularly beneficial for tasks that require understanding complex
dependencies in sequences, such as natural language processing.


Context:
3.2.2 Multi-Head Attention

Instead of performing a single attention function with dmodel-
dimensional keys, values and queries, we found it beneficial to
linearly project the queries, keys and values h times with different,
learned linear projections to dk, dk and dv dimensions, respectively.
On each of these projected versions of queries, keys and values we
then perform the attention function in parallel, yielding dv-
dimensional

‘To illustrate why the dot products get large, assume that the
components of q and k are independent random variables with mean 0 and
variance 1. Then their dot product, g -k = ves, qiki, has mean 0 and
variance dx.

4
```

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

MultiHead(Q, K, V) = Concat(head1, ..., headh)W O where headi = Attention(QW Q i , KW K i , V W V i)

Where the projections are parameter matrices W Q and W O ∈ Rhdv×dmodel. i ∈ Rdmodel×dk , W K i ∈ Rdmodel×dk , W V i ∈ Rdmodel×dv

In this work we employ h = 8 parallel attention layers, or heads. For each of these we use dk = dv = dmodel/h = 64. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

• In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].

• The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.

• Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to −∞) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

----

big

6

1024

4096

16

0.3

300K 4.33

26.4

development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size dk hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with dmodel = 1024 on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we

9

213

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)
Page number: 9

----

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In Advances in Neural Information Processing Systems, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. CoRR, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers), pages 434–443. ACL, August 2013.

Attention Visualizations

2 i i= RE 3 2 i 2 = = 2c 3 2 £ om % S GBANAAAA FS fe} oD DOD *H o Poe Q €oe2s oyzveyzuyeys 2SE @T_ EFESeSsEzESHR PL, SSSe TZSsSsggsg S=@LEGDEwB ECC oF aAeC RGN ESLSSSEESC -.VvVVVV VV HMO KEBOCSRSHLHOD QKXRDXE EO “A AAAAAA “= <2 £ 8 FogesouggsS ss P25 5273 Qvryxapvs\3 es sa 5 Seeneteecorzgrs 2g ogs aaa oO 2 Sere =~ aA o ° 8 ueeeogoa0 o © £ 5 ane) vvvvv Vv £ eg € ° 2 Ss v Do <¢ 8 & |

Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6.

Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

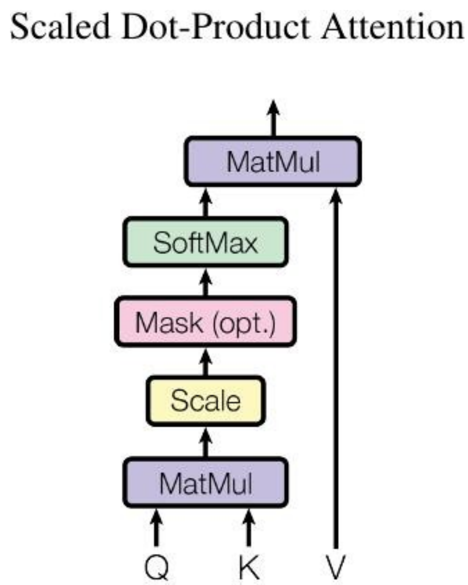
<ped> <ped> UOIUIdO == Aw ul Bulssiw ale « aM = yeum = S| sy ysnf pinoys = uoluldo Aw ul Bulssiw ae ysnf 38q Pinoys uojeojdde si! nq poped 38q JaAou Me] au <ped> <SOa> uojuido Aw ul Bulssiuw oe aM yeum S| SIU} ysnf 3q Pinoys uojeodde Ss}! ynq yoped 3q 4eAeuU meq auL <ped> <SOa> uo|uldo Aw ul Bulssiuw oe eM yeum S| Siu} ysnf 3q Pinoys uoyeoydde si! ynq yoped 3q aul

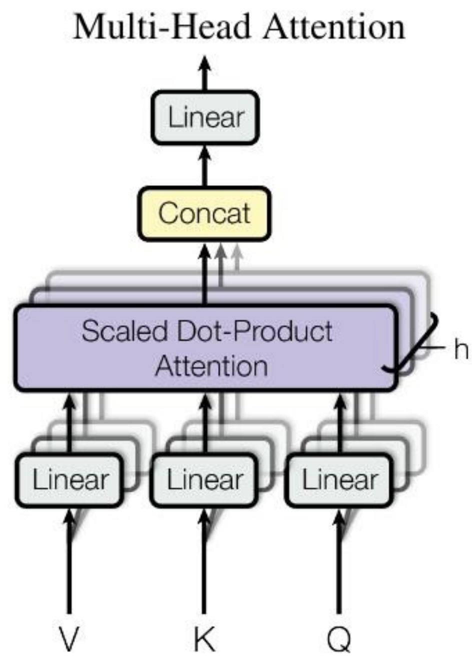
Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

<ped> <ped> <SOH>\ <SO3> uoluido = uoluido Aw Aw yeyum S| sim pi—f}— 4 -ysn{ | Pinoys «+ pinoys uoeoydde uojeodde SHI S$}! | ya | nga ynq poped pooped aq aq Janou™ J@AoU WIM TIM me) me) aul “OU

<ped> <ped> so <0 Uo|UIdO uoluido Aw Aw ul ul Bulssiw Bulssiw ae ale aM am yeuM yeum S| S| sty} # sly -—A - el eq eq pinoys « pinoys uojeoidde ee Ss}! Ss}! nq ee popod a —_ ee eq eq JOAoU JOAoU IW IW rr auL auL

Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.





References

- LangChain Inspiration
- Multivector Storage