EE 604
Dhruv - 210338

Q1

Code -

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load image
image = cv2.imread('rain.png')

threshold = 140
max_search_radius = 20

def is_above_threshold(pixel):
    return np.all(pixel > threshold)

processed_image = image.copy()

# Black out pixels above the threshold
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if is_above_threshold(image[i, j]):
            processed_image[i, j] = [0, 0, 0]

black_processed_image = processed_image.copy()

def black_out_neighbors(image, i, j):
    for di in range(-3, 4):
        for dj in range(-3, 4):
            ni, nj = i + di, j + dj
            if 0 <= ni < image.shape[0] and 0 <= nj < image.shape[1]:
                image[ni, nj] = [0, 0, 0]

for i in range(processed_image.shape[0]):
    for j in range(processed_image.shape[1]):
        if np.array_equal(processed_image[i, j], [0, 0, 0]):
            black_out_neighbors(black_processed_image, i, j)

cv2.imwrite('processed_image_with_black_boundary.png',
black_processed_image)
```

```
def find_nearest_non_threshold_pixel(image, x, y, threshold=140,
max_search_radius=10):
    min_distance = float('inf')
    closest_pixel = None

    for direction in ["left", "right", "up", "down"]:
        for radius in range(1, max_search_radius + 1):
            nx, ny = (x - radius, y) if direction == "left" else \
                     (x + radius, y) if direction == "right" else \
                     (x, y - radius) if direction == "up" else \
                     (x, y + radius)

            if 0 <= nx < image.shape[1] and 0 <= ny < image.shape[0]:
                r, g, b = image[ny, nx]
                if r > 0 and g > 0 and b > 0:
                    if radius < min_distance:
                        min_distance = radius
                        closest_pixel = image[ny, nx]

    return closest_pixel if closest_pixel is not None else image[y, x]

boundary_image = black_processed_image.copy()
for i in range(black_processed_image.shape[0]):
    for j in range(black_processed_image.shape[1]):
        if np.array_equal(black_processed_image[i, j], [0, 0, 0]):
            boundary_image[i, j] =
find_nearest_non_threshold_pixel(black_processed_image, j, i, threshold,
max_search_radius)

cv2.imwrite('boundary.png', boundary_image)

# Apply Gaussian blur
smoothed_image = cv2.GaussianBlur(boundary_image, (7, 7), 0)
cv2.imwrite('smoothed_boundary_image.png', smoothed_image)

# Plot images
plt.figure(figsize=(10, 10))

plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

plt.subplot(1, 2, 2)
plt.title('Smoothed Processed Image')
plt.imshow(cv2.cvtColor(smoothed_image, cv2.COLOR_BGR2RGB))
```

```
plt.show()
```

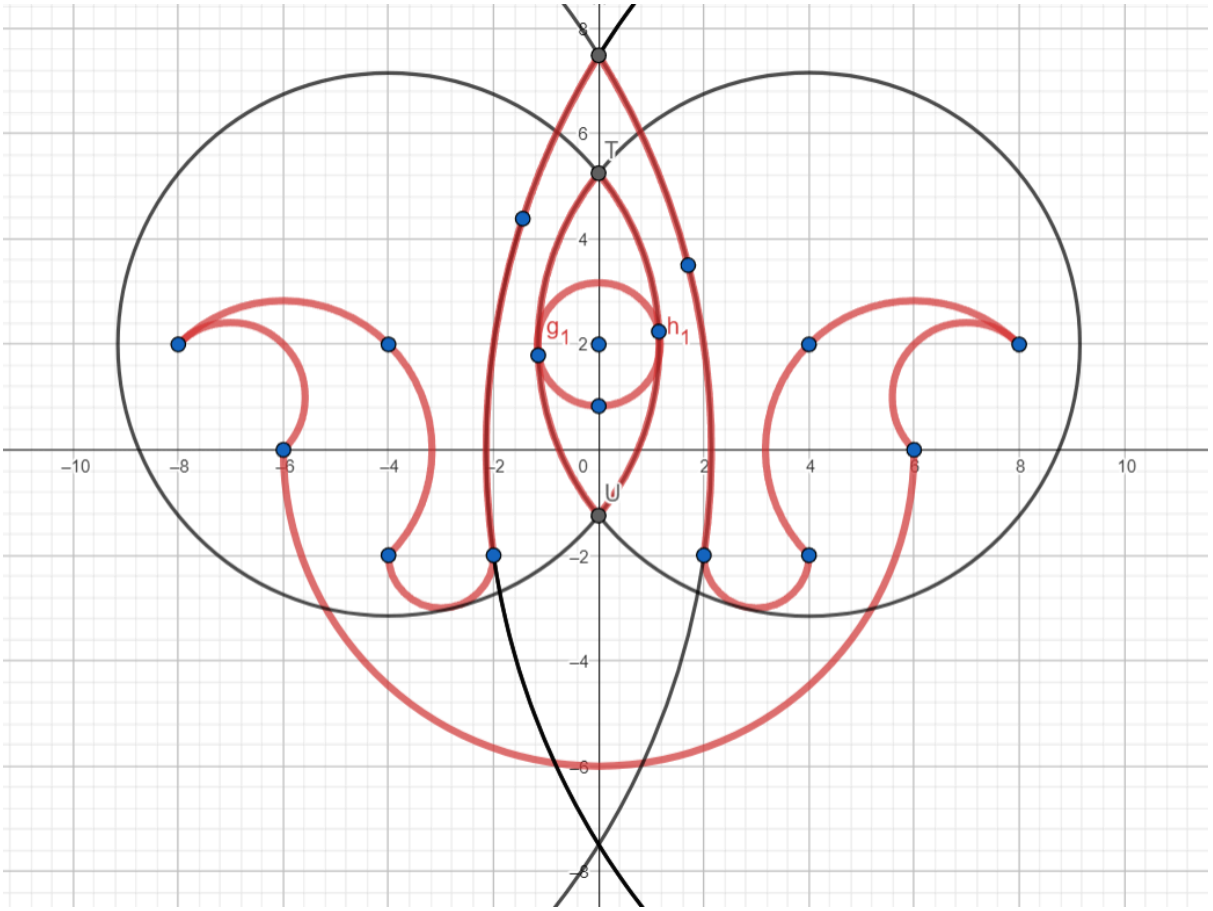
Image -

Q2

Used **matplotlib** to create a visual representation of the IIT Kanpur logo. Used multiple semicircles and wedges arranged in a specific pattern to draw inner part of the logo and added text in both English and Hindi around the central design. The resulting image is saved as a PNG file and displayed using matplotlib.

The PNG obtained was later converted to Binary Format.

The shapes dimensions were decided based on the GeoGebra design



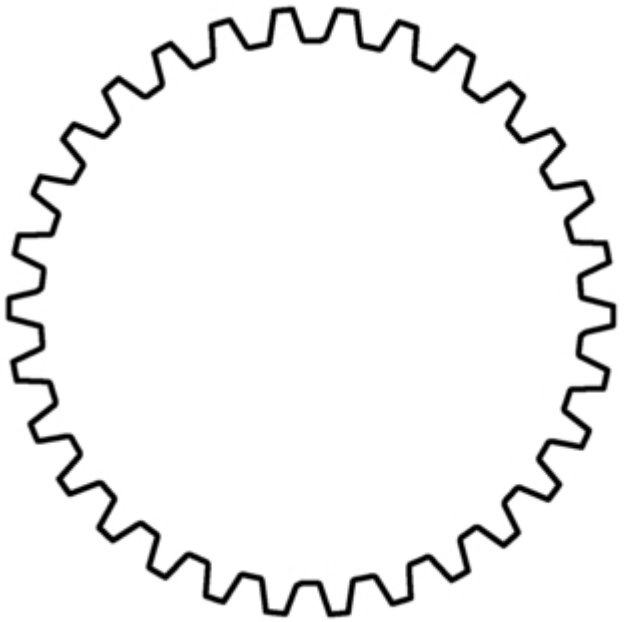
Code -

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
from matplotlib.font_manager import FontProperties

# Create a figure and axis
fig, ax = plt.subplots(figsize=(4, 4))  # Set figure size to 4x4 inches

# Define canvas size and center
canvas_size = 50
center_x, center_y = canvas_size // 2, canvas_size // 2

# Define radius values
radii = {
    'r1': 6,
    'r2': np.sqrt(2),
    'r3': 2 * np.sqrt(2),
    'r4': 1,
    'r5': 14.1,
    'r6': 5,
    'r7': 10,
    'r8': 14,
    'r9': 20
}

# Create semicircles
semicircles = [
    patches.Arc((center_x, center_y), 2 * radii['r1'], 2 * radii['r1'],
angle=-180, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x - 7, center_y + 1), 2 * radii['r2'], 2 *
radii['r2'], angle=-45, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x + 7, center_y + 1), 2 * radii['r2'], 2 *
radii['r2'], angle=45, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x + 6, center_y), 2 * radii['r3'], 2 * radii['r3'],
angle=45, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x - 6, center_y), 2 * radii['r3'], 2 * radii['r3'],
angle=-45, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x - 3, center_y - 2), 2 * radii['r4'], 2 *
radii['r4'], angle=180, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x + 3, center_y - 2), 2 * radii['r4'], 2 *
radii['r4'], angle=-180, theta1=0, theta2=180, edgecolor='black'),
    patches.Arc((center_x + 12, center_y), 2 * radii['r5'], 2 * radii['r5'],
angle=0, theta1=148, theta2=-171, edgecolor='black'),
    patches.Arc((center_x - 12, center_y), 2 * radii['r5'], 2 * radii['r5'],
```

```
angle=0, theta1=351, theta2=32, edgecolor='black'),
    patches.Wedge((center_x, center_y + 2), 1, 0, 360, facecolor='black',
edgecolor='black'),
    patches.Arc((center_x - 4, center_y + 2), 2 * radii['r6'], 2 *
radii['r6'], angle=0, theta1=322, theta2=37, edgecolor='black'),
    patches.Arc((center_x + 4, center_y + 2), 2 * radii['r6'], 2 *
radii['r6'], angle=0, theta1=144, theta2=-143, edgecolor='black'),
    patches.Arc((center_x, center_y), 2 * radii['r7'], 2 * radii['r7'],
angle=0, theta1=0, theta2=360, edgecolor='black'),
    patches.Arc((center_x, center_y), 2 * radii['r8'], 2 * radii['r8'],
angle=0, theta1=0, theta2=360, edgecolor='black'),
    patches.Arc((center_x, center_y), 2 * radii['r9'], 2 * radii['r9'],
angle=0, theta1=0, theta2=360, edgecolor='black'),
    patches.Wedge((center_x - 17, center_y + 3), 1, 0, 360,
facecolor='black', edgecolor='black'),
    patches.Wedge((center_x + 17, center_y + 3), 1, 0, 360,
facecolor='black', edgecolor='black'),
    patches.Wedge((center_x + 4.5, center_y + 0.5), 0.5, 0, 360,
facecolor='black', edgecolor='black'),
    patches.Wedge((center_x - 4.5, center_y + 0.5), 0.5, 0, 360,
facecolor='black', edgecolor='black')
]

# Add semicircles to the plot
for semicircle in semicircles:
    ax.add_patch(semicircle)

# Add vertical and horizontal lines
lines = [
    patches.ConnectionPatch((center_x - 1.5, center_y - 6), (center_x - 1.5,
center_y - 7.5), "data", "data", color="black", linewidth=1),
    patches.ConnectionPatch((center_x + 1.5, center_y - 6), (center_x + 1.5,
center_y - 7.5), "data", "data", color="black", linewidth=1),
    patches.ConnectionPatch((center_x - 1.5, center_y - 7.5), (center_x +
1.5, center_y - 7.5), "data", "data", color="black", linewidth=1)
]

for line in lines:
    ax.add_patch(line)

# Add text around the circle
text = "INDIAN INSTITUTE OF TECHNOLOGY"
n_chars = len(text)
angle_step = 180 / n_chars
text_radius = radii['r8'] + 4

for i, char in enumerate(text):
    angle = np.deg2rad(i * angle_step - 180)
```

```
    x = center_x + text_radius * np.cos(angle)
    y = center_y + text_radius * np.sin(angle)
    rotation_angle = np.degrees(angle) + 90
    ax.text(x, y, char, ha='center', va='center', fontsize=10,
rotation=rotation_angle, rotation_mode='anchor')

# Add text in Devanagari script
text = "भारतीय परदयोगीता संस्थान कानपुर"
words = text.split()
total_chars = sum(len(word) for word in words)
angle_step = 130 / total_chars
text_radius = radii['r8'] + 3

font_prop = FontProperties(fname='NotoSansDevanagariUI-Regular.ttf',
size=10)
current_angle = 155

for word in words:
    word_angle_step = angle_step * len(word)
    word_angle = np.deg2rad(current_angle - word_angle_step / 2)
    x = center_x + text_radius * np.cos(word_angle)
    y = center_y + text_radius * np.sin(word_angle)
    rotation_angle = np.degrees(word_angle) - 90
    ax.text(x, y, word, fontproperties=font_prop, ha='center', va='center',
fontsize=10, rotation=rotation_angle, rotation_mode='anchor')
    current_angle -= word_angle_step

# Set plot limits and aspect ratio
ax.set_xlim(center_x - radii['r9'] - 10, center_x + radii['r9'] + 10)
ax.set_ylim(center_y - radii['r9'] - 10, center_y + radii['r9'] + 10)
ax.set_aspect('equal')
ax.set_xticks(np.arange(0, canvas_size, 10))
ax.set_yticks(np.arange(0, canvas_size, 10))
ax.grid(which='both', linestyle='--', linewidth=0.5)
ax.axis('off')

# Save the figure to a file
file_path = 'IITK_LOGO.png'
fig.savefig(file_path)

# Show the plot
plt.show()
```

Took the gear from some other image and added it on to IITK Logo by iterating over the pixel values in binary format
https://images.app.goo.gl/EyyepDmvpNuhUacz5




भारतीय प्रदयोगीता संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY