Name: Prasad Kute

Roll No: 331034

Div: A **Batch:** A1 Sub: Artificial Intelligence

Assignment No – 5

Aim: Implement graph coloring problem. OR Implementation of Constraint Satisfaction Problem for solving Crypt-arithmetic Problems. Solve any one of the follow: Backtracking: Constraint Satisfaction Problem – 1) Seating Arrangement or Computer CPU utilization 2) Analyze data from twitter or news to categorize them useful reading list out of the whole clutter 3) Read data from images using OCR (optical character recognition) and OMR (optical mark recognition) techniques – use open source libs

Source Code:

```
import warnings
import cv2
import easyocr
import numpy as np
from google.colab.patches import cv2_imshow  # Import cv2_imshow for Colab

warnings.filterwarnings("ignore", category=FutureWarning, module="torch")

def preprocess_image(image_path):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)
    return image, gray, thresh

def perform_ocr(image):
    try:
        reader = easyocr.Reader(['en'], gpu=True)
    except Exception as e:
        print(f"Error initializing EasyOCR with GPU: {e}")
        print("Falling back to CPU.")
        reader = easyocr.Reader(['en'], gpu=False)

    result = reader.readtext(image)
    text = " ".join([res[1] for res in result])
    return text

def perform_omr(thresh):
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
```

```
cv2.CHAIN_APPROX_SIMPLE)
    marks = []
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        if w > 20 and h > 20:
            marks.append((x, y, w, h))
    return marks

def main(image_path):
    image, gray, thresh = preprocess_image(image_path)

    ocr_text = perform_ocr(image)
    print("OCR Text:")
    print(ocr_text)

    marks = perform_omr(thresh)
    print("OMR Marks:")
    for mark in marks:
        x, y, w, h = mark
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Use cv2_imshow() instead of cv2.imshow()
    cv2_imshow(image)  # Display the image with OMR marks

if __name__== "__main__":
    image_path = '/content/image.jpg'
    main(image_path)
```

Output:

OCR Text:

Tesseract sample

OMR Marks:

```
OCR Text:
Tesseract sample
OMR Marks:
```