```
import re
from datetime import datetime, timedelta


MONTH_MAP = {
    'Jan': '01', 'Feb': '02', 'Mar': '03', 'Marc': '03', 'Apr': '04', 'May': '05',
'Jun': '06',
    'Jul': '07', 'Aug': '08', 'Sep': '09', 'Oct': '10', 'Nov': '11', 'Dec': '12',
    'January': '01', 'February': '02', 'March': '03', 'April': '04', 'June': '06',
    'July': '07', 'August': '08', 'September': '09', 'October': '10', 'November':
'11', 'December': '12'
}

MONTH_MAP_INVERSE_ABBR = {v: k for k, v in MONTH_MAP.items() if len(k) <= 3}


DATE_PATTERNS = [
    r'\b(\d{1,2})\s*(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec|January|
February|March|April|May|June|July|August|September|October|November|December)[a-
z]*\.?,?\s*(\d{4})',
    r'\b(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec|January|February|March|
April|May|June|July|August|September|October|November|December)[a-z]*\.?,?\s*(\
d{1,2})?,?\s*(\d{4})',
    r'(\d{1,2})\s*(Jan|Feb|Mar|Marc|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[a-
z]*\.?,?\s*(\d{2,4})',
    r'(Jan|Feb|Mar|Marc|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec|January|February|March|
April|May|June|July|August|September|October|November|December)[a-z]*\.?,?\s*(\
d{4})',
    r'(Jan|Feb|Mar|Marc|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[a-z]*\.?,?\s*(\
d{1,2})?,?\s*(\d{2,4})',
    r'(January|February|March|April|May|June|July|August|September|October|
November|December)\s*\.?,?\s*(\d{1,2})?,?\s*(\d{2,4})',
    r'(Jan|Feb|Mar|Marc|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec|January|February|March|
April|May|June|July|August|September|October|November|December)[a-z]*\.?,?\s*(\
d{2,4})',
    r'(\d{1,2})[/-](\d{1,2})[/-](\d{2,4})',
    r'(\d{1,2})[/-](\d{4})',
    r'(\d{4})'
]

def _parse_date_groups(groups):
    if len(groups) == 3:
        if groups[0] in MONTH_MAP:
            month, day, year = groups
            month = MONTH_MAP[month]
        elif groups[1] in MONTH_MAP:
            day, month, year = groups
            month = MONTH_MAP[month]
        else:
            month, day, year = groups
    elif len(groups) == 2:
        if groups[0] in MONTH_MAP:
            month, year = groups
            month = MONTH_MAP[month]
            day = '1'
        elif groups[0].isdigit():
            month, year = groups
            day = '1'
        else:
            year, month = groups
            month = MONTH_MAP[month]
            day = '1'
    else:
        year = groups[0]
        month, day = '1', '1'

    year = int(year)
    if year < 100:
        year += 1900

    return year, month, day

def normalize_date(date_str):
    date_str = re.sub(r'\b\d{3}-\d{3}-\d{4}\b', '', date_str)

    for pattern in [DATE_PATTERNS[0], DATE_PATTERNS[1]]:
        match = re.search(pattern, date_str)
        if match:
            groups = match.groups()
            if pattern == DATE_PATTERNS[0]:
                day, month, year = groups
            else:
                month, day, year = groups
                day = day if day else '01'

            month = MONTH_MAP[month[:3]] if month else None
            if month is None:
                continue

            return f"{year}-{month}-{int(day):02d}"

    for pattern in DATE_PATTERNS[2:]:
        match = re.search(pattern, date_str)
        if match:
            year, month, day = _parse_date_groups(match.groups())
            try:
                date = datetime(year, int(month), int(day))
                return date.strftime('%Y-%m-%d')
            except ValueError:
                continue

    return None

def denormalize_date(normalized_date):
    year, month, day = normalized_date.split('-')
    return f"{MONTH_MAP_INVERSE_ABBR[month]} {int(day):02d}, {year}"

def process_file(input_file, output_file):
    with open(input_file, 'r') as f_in, open(output_file, 'w') as f_out:
        for line in f_in:
            line_number, content = line.strip().split('\t', 1)
            date = normalize_date(content)
            if date:
                pushed_date = (datetime.strptime(date, '%Y-%m-%d') +
timedelta(days=40)).strftime('%Y-%m-%d')
                original_format_pushed = denormalize_date(pushed_date)
                f_out.write(f"{line_number}\t{date}\t{pushed_date}\
t{original_format_pushed}\n")
```

```
process_file('WN25_dates.txt', 'LHS712-Assg1-DMEGHANA.txt')
```