In [41]:
```
#Problem 1
class binomial_coefficients:

    def get_n(self, n):
        assert n>=0, "Binominal coefficients not defined for negative number."

        #initialize a list for binomial coefficients for all lines
        binomial = []

        for i in range(n+1):
            #initialize an empty list for each line
            binomial.append([])

            for k in range(i+1):
                #the first number and the last number in each line are 1
                if k == 0 or k == i:
                    binomial[i].append(1)
                #k-th number in i-th line = (k-1)th number in (i-1)th line + k-th number in (i-1)th line
                else:
                    binomial[i].append(binomial[i-1][k-1]+binomial[i-1][k])

        return binomial[n]

    def get_nk(self, n, k):
        assert n>=0, "Binominal coefficients not defined for negative number."
        assert k>=0 and k<=n, "Binominal coefficients not defined for k<0 or k>n."

        if k == 0 or k == n:
            return 1
        else:
            return self.get_n(n)[k]

    def print_pt(self,n):
        last_row = self.get_n(n)
        ele_width = len(str(max(last_row))) + 1 #space for the longest element
        bottom_width = ele_width * len(last_row) #the width of the longest line


        for i in range(n+1):
            row = "" #initialize a string to store each line

            for j in range(i+1):
                number_nk = str(self.get_nk(i,j))
                row += number_nk  + " " * (ele_width-len(number_nk)) #add the (n,k) element and use spaces
                                                                    #so that each element occupies the same length

            print(row.center(bottom_width))
```

In [42]:
```
b = binomial_coefficients()
b.get_n(1)
b.get_n(4)
```

Out[42]:
```
[1, 4, 6, 4, 1]
```

In [43]:
```
b.get_nk(4,3)
```

Out[43]:
```
4
```

In [44]:
```
b.print_pt(2)
```

```
  1
 1 1
1 2 1
```

In [45]:
```
b.print_pt(5)
```

```
      1
    1   1
  1   2   1
 1   3   3   1
1   4   6   4   1
```

In [46]:
```
b.print_pt(12)
```

```
                      1
                   1     1
                1     2     1
             1     3     3     1
          1     4     6     4     1
       1     5    10    10     5     1
    1     6    15    20    15     6     1
 1     7    21    35    35    21     7     1
   1     8    28    56    70    56    28     8     1
1     9    36    84   126   126    84    36     9     1
   1    10    45   120   210   252   210   120    45    10     1
1    11    55   165   330   462   462   330   165    55    11     1
   1    12    66   220   495   792   924   792   495   220    66    12     1
```

In [59]:
```
#Problem 2
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

#read in csv data
df=pd.read_csv('owid-covid-data.csv')
#Convert argument to datetime
df["date"]= pd.to_datetime(df.date)
df.head()
```

Out[59]:

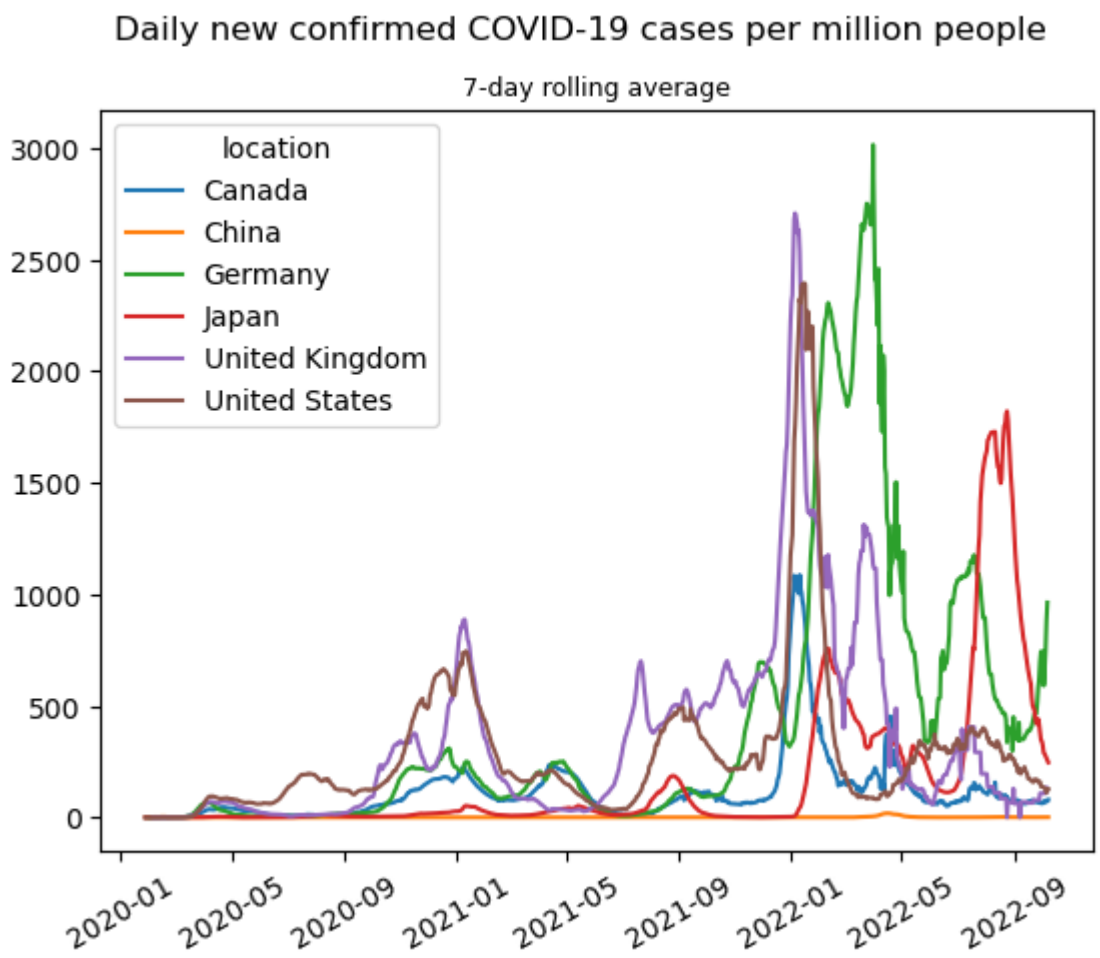
	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	NaN	NaN	NaN
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	NaN	NaN	NaN
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	NaN	NaN	NaN
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	NaN	NaN	NaN
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	NaN	NaN	NaN

5 rows × 67 columns

In [80]:
```
#pick 6 countries and plot the chart of new confirmed COVID cases per country.
countries = df[df['location'].isin(['China','United Kingdom','United States','Japan','Canada','Germany'])]
countries
chart = sns.lineplot(data=countries,x ='date', y='new_cases_smoothed_per_million',hue ='location')

plt.suptitle("Daily new confirmed COVID-19 cases per million people",fontsize=12)
plt.title("7-day rolling average",fontsize=9)
plt.xticks(rotation = 30)

chart.set(xlabel=None,ylabel=None)
plt.show()
```



In [94]:
```
#Plot the world map with current cases (pick a date from Sep 2022)
current_cases = df[df['date']=='2022-09-17']
fig1 = px.choropleth(current_cases, locations="location",
                     locationmode = "country names",
                     color="total_cases_per_million",
                     color_continuous_scale=px.colors.sequential.Oranges,
                     title="Daily new confirmed COVID-19 cases per million people, Sep 17, 2022")
fig1.show()
```

In [96]:
```
#Plot with cases from a date in 2020, for each country.
previous_cases = df[df['date']=='2020-09-17']
fig2 = px.choropleth(previous_cases, locations="location",
                     locationmode = "country names",
                     color="total_cases_per_million",
                     color_continuous_scale=px.colors.sequential.Oranges,
                     title="Daily new confirmed COVID-19 cases per million people, Sep 17, 2020")
fig2.show()
```

In [ ]: