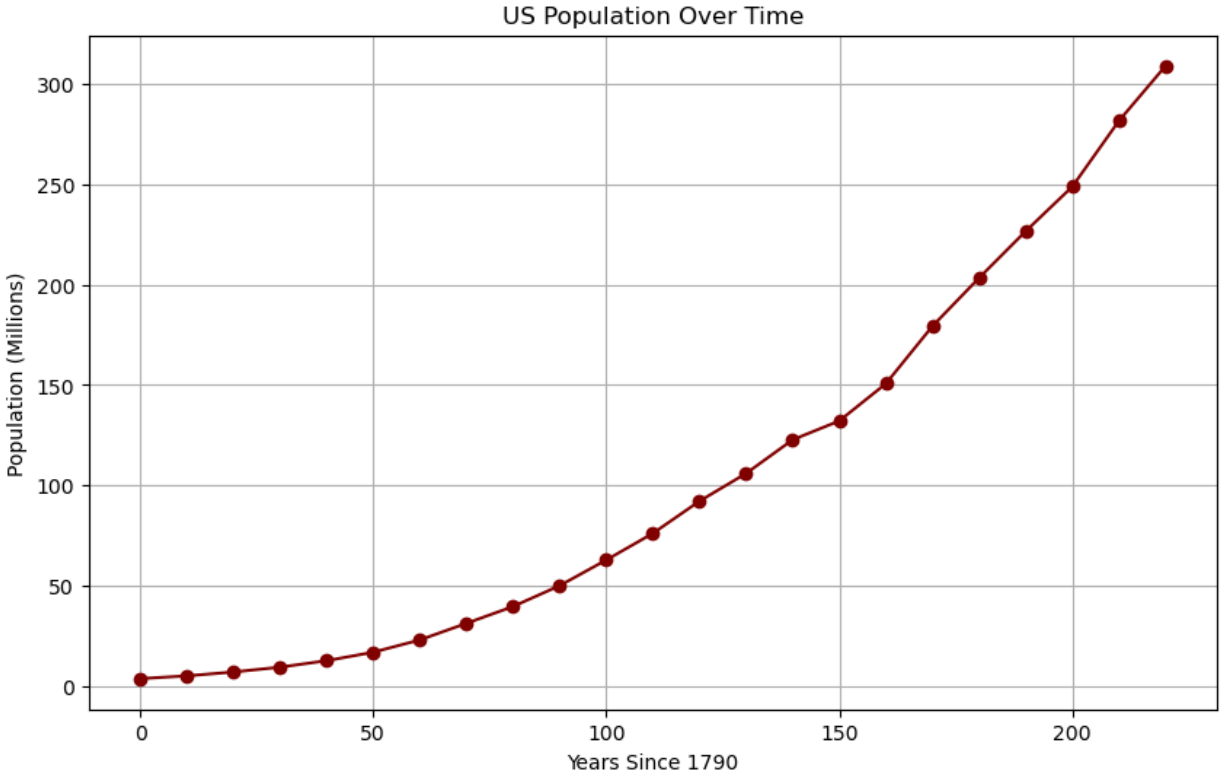
In [2]:
```
#Import the data and create two new columns. Create one column that is the number of y
#us pop CSV.csv

#import pd
import pandas as pd

#load the csv
data = pd.read_csv('us pop CSV.csv')

#one column is years, the other column is population
#years

data["years since 1790"] = data['year'] - 1790

#population
data['us_pop (millions)'] = data['us_pop'] / 1000000
print(data)
```

```
    year     us_pop  years since 1790  us_pop (millions)
0   1790    3929326                 0           3.929326
1   1800    5308483                10           5.308483
2   1810    7239881                20           7.239881
3   1820    9638453                30           9.638453
4   1830   12866020                40          12.866020
5   1840   17069453                50          17.069453
6   1850   23191876                60          23.191876
7   1860   31443321                70          31.443321
8   1870   39818449                80          39.818449
9   1880   50189209                90          50.189209
10  1890   62947714               100          62.947714
11  1900   76212168               110          76.212168
12  1910   92228496               120          92.228496
13  1920  106021537               130         106.021537
14  1930  122775046               140         122.775046
15  1940  132164569               150         132.164569
16  1950  150697361               160         150.697361
17  1960  179323175               170         179.323175
18  1970  203302031               180         203.302031
19  1980  226545805               190         226.545805
20  1990  248709873               200         248.709873
21  2000  281421906               210         281.421906
22  2010  308745538               220         308.745538
```

In [2]:
```
#Plot the US population (in millions) versus the years since 1790.


#import pd, plt
import pandas as pd
import matplotlib.pyplot as plt

#load in csv
data = pd.read_csv('us pop CSV.csv')

#column for number of years since 1790
data['Years Since 1790'] = data['year'] - 1790

#column for population in millions
data['us_pop (Millions)'] = data['us_pop'] / 1000000
```

```
#data plot
plt.figure(figsize=(10, 6))
plt.plot(data['Years Since 1790'], data['us_pop (Millions)'], color = 'maroon' ,marker
plt.title('US Population Over Time')
plt.xlabel('Years Since 1790')
plt.ylabel('Population (Millions)')
plt.grid(True)
plt.show()
```

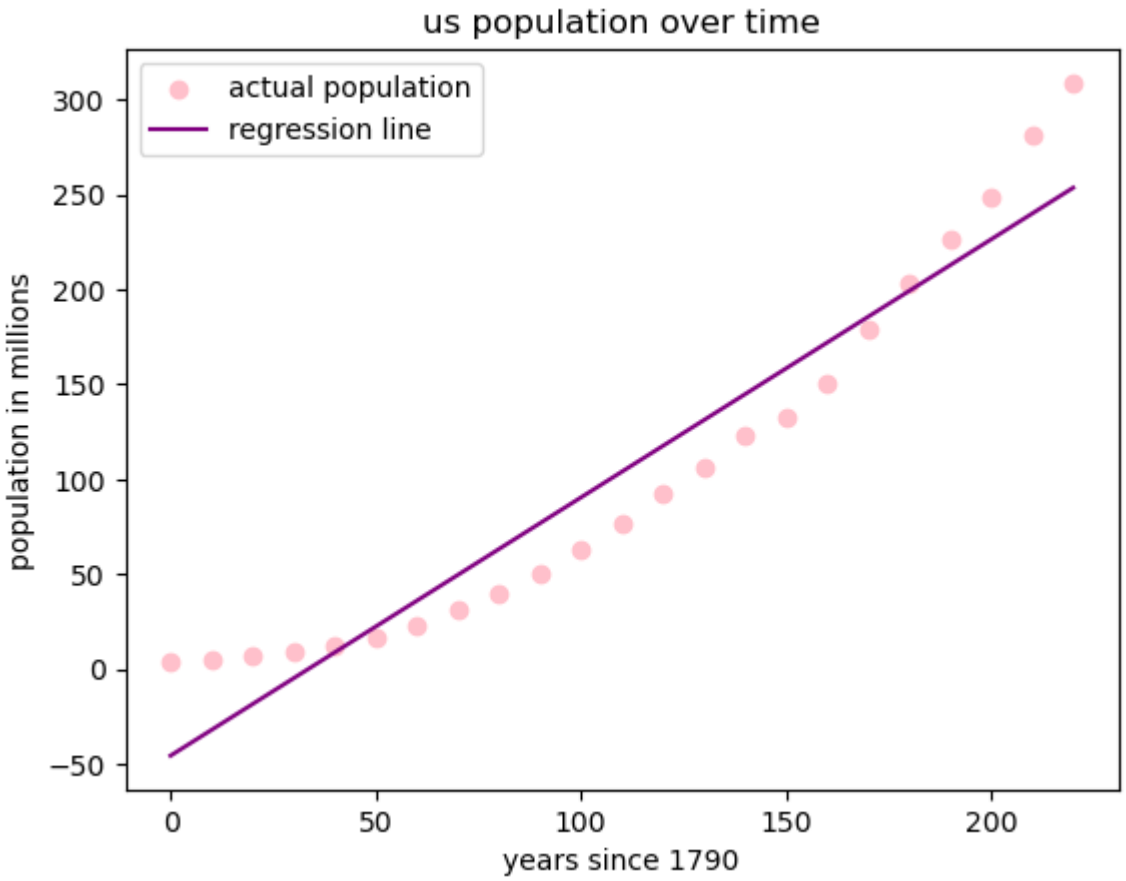


In [24]:

```
#Create a linear regression model to predict the US population (in millions) t years f

#load pd, sklearn,plt, r2, LR
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

#load csv
#us pop CSV.csv
data = pd.read_csv('us pop CSV.csv')


#column for the number of years since 1790
#column for population in millions
data['years since 1790'] = data['year'] - 1790
data['population in millions'] = data['us_pop'] / 1e6


#define x and y
X = data[['years since 1790']]
y = data['population in millions']

#LR model
```

```
model = LinearRegression()
model.fit(X, y)

#pop predict
y_pred = model.predict(X)

#r2 value
r_squared = r2_score(y, y_pred)
print(f'R-squared value: {r_squared:.4f}')

#plot data
plt.scatter(X, y, label='actual population', color='pink')
plt.plot(X, y_pred, color='purple', label='regression line')
plt.xlabel('years since 1790')
plt.ylabel('population in millions')
plt.title('us population over time')
plt.legend()
plt.show()
```

```
R-squared value: 0.9192
```



In [25]:

```
#Create another new column in your data by squaring the number of years since 1790.
#load pd
import pandas as pd


#load csv
#us pop CSV.csv
data = pd.read_csv('us pop CSV.csv')

#new column years since 1790
data['years since 1790'] = data['year'] - 1790
```

```
#new years squared column
data['years squared'] = data['years since 1790'] ** 2
print(data.head())
```

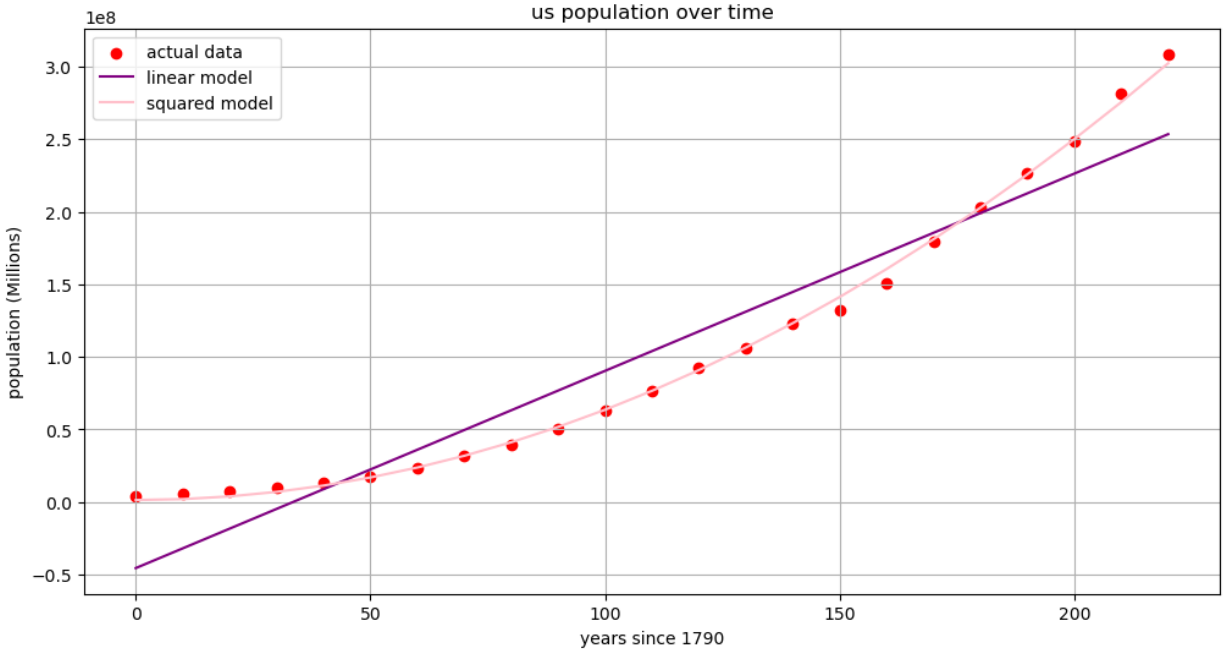
```
   year    us_pop  years since 1790  years squared
0  1790   3929326                 0              0
1  1800   5308483                10            100
2  1810   7239881                20            400
3  1820   9638453                30            900
4  1830  12866020                40           1600
```

In [31]:
```
#Run another linear regression, where your input feature is the square of the number c

#load pd, sklearn,LR, r2
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

#load csv
#us pop CSV.csv
data = pd.read_csv('us pop CSV.csv')

#new column years since 1790
data['Years Since 1790'] = data['year'] - 1790

#new years squared column
data['Years Squared'] = data['Years Since 1790'] ** 2

#define x and y
X = data[['Years Squared']]
y = data['us_pop']

#LR model
model = LinearRegression()
model.fit(X, y)

#predict
y_pred = model.predict(X)

#r2
r_squared = r2_score(y, y_pred)
print(f'R-squared value: {r_squared:.4f}')
```

```
R-squared value: 0.9985
```

In [36]:
```
#Plot the models you built on top of the data. Which one fits the data better? Is this

#load pd, sklearn,LR, r2
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np

#load csv
#us pop CSV.csv
data = pd.read_csv('us pop CSV.csv')


#new column years since 1790
```

```
data['years since 1790'] = data['year'] - 1790

#new years squared column
data['years squared'] = data['years since 1790'] ** 2

#define x , y
X = data[['years since 1790']]
y = data['us_pop']

#make LR model
model_linear = LinearRegression()
model_linear.fit(X, y)

#LR model with squared
X_squared = data[['years squared']]
model_squared = LinearRegression()
model_squared.fit(X_squared, y)

#predict
y_pred_linear = model_linear.predict(X)

#predict sqrd model
y_pred_squared = model_squared.predict(X_squared)

#LR model
plt.figure(figsize=(12, 6))
plt.scatter(data['years since 1790'], data['us_pop'], label='actual data', color ='red
plt.plot(data['years since 1790'], y_pred_linear, label='linear model', color ='purple
plt.plot(data['years since 1790'], y_pred_squared, label='squared model', color ='pink

plt.title('us population over time')
plt.xlabel('years since 1790')
plt.ylabel('population (Millions)')
plt.legend()
plt.grid(True)

plt.show()
```



The squared model fits the data better.