```
In [25]:  import pandas as pd
```

Calculation for Morgage 2

```
In [94]:  principal = 1000000 # 1million dollars
          interest_rate = 2.5 # annual interest provided for mortgage 2
          years = 20 # loan term
          monthly_interest_rate = interest_rate/12
          period = years*12
```

Above function calculate the EMI (fixed amount) amount that will be paid on monthly basis till the loan term ends.

```
In [83]:  def calculate_fixed_amount(principal, interest_rate, period):
              r = (1 + interest_rate/100) ** period
              return principal * (interest_rate/100 * r) / (r - 1)
```

```
In [84]:  amort_amount = calculate_fixed_amount(principal, monthly_interest_rate, period)
```

```
In [85]:  amort_amount
```

```
Out[85]:  5299.028930322097
```

Declaring an pandas dataframe with required columns and intializing it with value when loan was given to the mortgagee from bank

```
In [86]:  amort_schedule_df = pd.DataFrame( {'Month Number' : [0],
                                             'Beginning Balance ($)': [""],
                                             'Fixed Payment Amount ($)': [""],
                                             'Principal Paydown ($)' : [""],
                                             'Interest Applied ($)' : [""],
                                             'Ending Balance ($)' : [principal]
                                            })
```

```
In [87]:  amort_schedule_df
```

Out[87]:

	Month Number	Beginning Balance ($)	Fixed Payment Amount ($)	Principal Paydown ($)	Interest Applied ($)	Ending Balance ($)
0	0					1000000

```
In [88]:  def calculate_amortization_schedule (amort_schedule_df,amort_amount, monthly_interest_rate, period):

              for p in range(1, period+1):
                  # p=1 check in another shell if below line works correctly
                  #amort_schedule_df.loc[p,'Beginning Balance'] = amort_schedule_df.loc[p-1,'Ending Balance']

                  amort_schedule_df.loc[p,'Beginning Balance ($)'] = amort_schedule_df.loc[p-1,'Ending Balance ($)']
                  amort_schedule_df.loc[p,'Month Number'] = int(p)
                  amort_schedule_df.loc[p, 'Fixed Payment Amount ($)'] = amort_amount
                  amort_schedule_df.loc[p, 'Interest Applied ($)'] =  amort_schedule_df.loc[p,'Beginning Balance ($)']*monthly_interest_rate/100
                  amort_schedule_df.loc[p, 'Principal Paydown ($)'] = amort_schedule_df.loc[p, 'Fixed Payment Amount ($)'] - amort_schedule_df.loc[p, 'Interest Applied ($)']
                  amort_schedule_df.loc[p,'Ending Balance ($)'] = amort_schedule_df.loc[p,'Beginning Balance ($)'] - amort_schedule_df.loc[p, 'Principal Paydown ($)']
                  # p = p+1

              return amort_schedule_df
```

```
In [89]:  amort_schedule_df = calculate_amortization_schedule (amort_schedule_df,amort_amount, monthly_interest_rate, period)
```

```
In [90]:  amort_schedule_df
```

Out[90]:

	Month Number	Beginning Balance ($)	Fixed Payment Amount ($)	Principal Paydown ($)	Interest Applied ($)	Ending Balance ($)
0	0.0					1.000000e+06
1	1.0	1000000	5299.02893	3215.695597	2083.333333	9.967843e+05
2	2.0	996784.304403	5299.02893	3222.394963	2076.633968	9.935619e+05
3	3.0	993561.90944	5299.02893	3229.108286	2069.920645	9.903328e+05
4	4.0	990332.801155	5299.02893	3235.835595	2063.193336	9.870970e+05
...	...	...	...	...	...	...
236	236.0	26330.35163	5299.02893	5244.174031	54.854899	2.108618e+04
237	237.0	21086.177599	5299.02893	5255.099394	43.929537	1.583108e+04
238	238.0	15831.078205	5299.02893	5266.047517	32.981413	1.056503e+04
239	239.0	10565.030688	5299.02893	5277.01845	22.010481	5.288012e+03
240	240.0	5288.012238	5299.02893	5288.012238	11.016692	6.353002e-08

241 rows × 6 columns

```
In [95]:  final_output = amort_schedule_df.applymap(lambda x : round(x,2) if type(x)!=str else x)
```

```
In [96]:  final_output
```

Out[96]:

	Month Number	Beginning Balance ($)	Fixed Payment Amount ($)	Principal Paydown ($)	Interest Applied ($)	Ending Balance ($)
0	0.0					1000000.00
1	1.0	1000000	5299.03	3215.7	2083.33	996784.30
2	2.0	996784.3	5299.03	3222.39	2076.63	993561.91
3	3.0	993561.91	5299.03	3229.11	2069.92	990332.80
4	4.0	990332.8	5299.03	3235.84	2063.19	987096.97
...	...	...	...	...	...	...
236	236.0	26330.35	5299.03	5244.17	54.85	21086.18
237	237.0	21086.18	5299.03	5255.1	43.93	15831.08
238	238.0	15831.08	5299.03	5266.05	32.98	10565.03
239	239.0	10565.03	5299.03	5277.02	22.01	5288.01
240	240.0	5288.01	5299.03	5288.01	11.02	0.00

241 rows × 6 columns

```
In [97]:  final_output.to_csv(r'C:\Users\nazne\Downloads\GWP1\amortization_schedule_mortgage2_new.csv', index = False)
          # path can be changed to directory where user want to save the final output
```

```
In [ ]:
```