This document is the Part 3 of the GWP 2 for Group 868 of MScFe 600

In [18]:
```
import pandas as pd
import numpy as np
from datetime import datetime
```

In [3]:
```
"""
We consider 3 types of mortgages here -

30 year fixed = type A
20 year fixed = type B
7 -1 ARM  = type C


We represent all 3 mortgages using a data frame , which is equivalent to  a
n excel table
"""
```

Out[3]:
```
'\nWe consider 3 types of mortgages here - \n\n30 year fixed = type A\n20 y
ear fixed = type B\n7 -1 ARM  = type C\n\n\nWe represent all 3 mortgages us
ing a data frame , which is equivalent to  an excel table\n'
```

In [4]:

```
def amortizer(loan_amount: int, interest_rate: float, loan_term: int) -> p
d.DataFrame:
    # fixed payment calculation
    monthly_interest_rate = interest_rate / 12
    number_of_payments = loan_term * 12
    fixed_payment = loan_amount * (monthly_interest_rate * (1 + monthly_int
erest_rate) ** number_of_payments) / (((1 + monthly_interest_rate) ** numbe
r_of_payments) - 1)

    # create empty dataframe
    amortization_schedule = pd.DataFrame(columns=["month_number", "fixed_pa
yment", "principal_pay_down", "interest_applied", "new_principal_balance"])

    # populate dataframe
    current_balance = loan_amount
    amortization_schedule = amortization_schedule.append({"month_number":
0, "fixed_payment": 0,
                                                          "principal_pa
y_down": 0, "interest_applied": 0,
                                                          "new_principa
l_balance": loan_amount}, ignore_index=True)
    for month in range(1, number_of_payments + 1):
        interest_applied = current_balance * monthly_interest_rate
        principal_pay_down = fixed_payment - interest_applied
        new_principal_balance = current_balance - principal_pay_down
        amortization_schedule = amortization_schedule.append({"month_numbe
r": month, "fixed_payment": fixed_payment,
                                                              "principal_pa
y_down": principal_pay_down, "interest_applied": interest_applied,
                                                              "new_principa
l_balance": new_principal_balance}, ignore_index=True)
        current_balance = new_principal_balance

    # display amortization schedule
    amortization_schedule.astype({'month_number':'int32'})
    amortization_schedule.set_index('month_number', drop=True, inplace= Tru
e )

    return amortization_schedule
```

In [5]:
```
# mortgage information for Mortgage A
loan_amount = 200000
interest_rate = 0.04
loan_term = 30


mortgageA = amortizer(loan_amount, interest_rate, loan_term)

mortgageA
```

Out[5]:

month_number	fixed_payment	principal_pay_down	interest_applied	new_principal_balance
0.0	0	0	0	200000
1.0	954.830591	288.163924	666.666667	199711.836076
2.0	954.830591	289.124471	665.70612	199422.711605
3.0	954.830591	290.088219	664.742372	199132.623386
4.0	954.830591	291.05518	663.775411	198841.568206
...	...	...	...	...
356.0	954.830591	939.074657	15.755934	3787.705631
357.0	954.830591	942.204905	12.625685	2845.500725
358.0	954.830591	945.345589	9.485002	1900.155137
359.0	954.830591	948.49674	6.33385	951.658396
360.0	954.830591	951.658396	3.172195	0.0

361 rows × 4 columns

In [6]:
```
mortgageA.sum()
```

Out[6]:
```
fixed_payment             343739.012735
principal_pay_down            200000.0
interest_applied          143739.012735
new_principal_balance    43121703.82054
dtype: object
```

Approximately 143.7k is the total interest paid over the 30 years and the monthly fixed payment is $955

In [7]:
```
# mortgage information for Mortgage B
loan_amount = 200000
interest_rate = 0.025
loan_term = 20


mortgageB = amortizer(loan_amount, interest_rate, loan_term)

mortgageB
```

Out[7]:

month_number	fixed_payment	principal_pay_down	interest_applied	new_principal_balance
0.0	0	0	0	200000
1.0	1059.805786	643.139119	416.666667	199356.860881
2.0	1059.805786	644.478993	415.326794	198712.381888
3.0	1059.805786	645.821657	413.984129	198066.560231
4.0	1059.805786	647.167119	412.638667	197419.393112
...	...	...	...	...
236.0	1059.805786	1048.834806	10.97098	4217.23552
237.0	1059.805786	1051.019879	8.785907	3166.215641
238.0	1059.805786	1053.209503	6.596283	2113.006138
239.0	1059.805786	1055.40369	4.402096	1057.602448
240.0	1059.805786	1057.602448	2.203338	0.0

241 rows × 4 columns

In [8]:
```
mortgageB.sum()
```

Out[8]:
```
fixed_payment              254353.388655
principal_pay_down             200000.0
interest_applied            54353.388655
new_principal_balance    26089626.554627
dtype: object
```

Approximately 54.3k is the total interest paid over the 20 years and the monthly fixed payment is ~$1060

7- 1 ARM

In [9]:
```
interest_rates = pd.read_csv('MORTGAGE30US.csv')
```

In [10]:
```
interest_rates
```

Out[10]:

	Unnamed: 0	DATE	MORTGAGE30US	Unnamed: 3	Unnamed: 4	Position	Rates
0	1	April 2, 1971	7.33	NaN	Start	1549.0	7.65
1	2	April 9, 1971	7.31	NaN	NaN	1550.0	7.54
2	3	April 16, 1971	7.31	NaN	NaN	1551.0	7.42
3	4	April 23, 1971	7.31	NaN	NaN	1552.0	7.17
4	5	April 30, 1971	7.29	NaN	NaN	1553.0	7.13
...	...	...	...	...	...	...	...
2667	2668	May 12, 2022	5.30	NaN	NaN	NaN	NaN
2668	2669	May 19, 2022	5.25	NaN	NaN	NaN	NaN
2669	2670	May 26, 2022	5.10	NaN	NaN	NaN	NaN
2670	2671	June 2, 2022	5.09	NaN	NaN	NaN	NaN
2671	2672	June 9, 2022	5.23	NaN	NaN	NaN	NaN

2672 rows × 7 columns

In [12]:
```
"""
# For the purpose of this exercise we chose from Jan 1 1991 to Dec 31 2020 as the 30 year period of this 7-1 ARM


## We further assume as follows,

The interest rate for the first 7 years will be the most recent interest rate as of Jan 1st 1991

The interest for each subsequent year will bt eh most recent interest rate of Jan 1st of the 1998, and so on
"""
```

Out[12]:
```
'\n# For the purpose of this exercise we chose from Jan 1 1991 to Dec 31 2020 as the 30 year period of this 7-1 ARM \n\n\n## We further assume as follows, \n\nThe interest rate for the first 7 years will be the most recent interest rate as of Jan 1st 1991\n\nThe interest for each subsequent year will bt eh most recent interest rate of Jan 1st of the 1998, and so on \n'
```

In [15]:
```
#now we calculate the interest rate for Year 1, Year 8 through Year 30 .

interest_rates.dtypes
```

Out[15]:
```
Unnamed: 0        int64
DATE             object
MORTGAGE30US    float64
Unnamed: 3      float64
Unnamed: 4       object
Position        float64
Rates           float64
dtype: object
```

In [16]:
```
interest_rates = interest_rates[['DATE', 'MORTGAGE30US']]
```

In [19]:
```
interest_rates.DATE = pd.to_datetime(interest_rates.DATE)
```

```
C:\Users\santh\AppData\Roaming\Python\Python37\site-packages\pandas\core\ge
neric.py:5516: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

In [20]:
```
interest_rates
```

Out[20]:

	DATE	MORTGAGE30US
0	1971-04-02	7.33
1	1971-04-09	7.31
2	1971-04-16	7.31
3	1971-04-23	7.31
4	1971-04-30	7.29
...	...	...
2667	2022-05-12	5.30
2668	2022-05-19	5.25
2669	2022-05-26	5.10
2670	2022-06-02	5.09
2671	2022-06-09	5.23

2672 rows × 2 columns

In [25]:
```
years = [1991] + list(range(1998,2021))
#we pick a 7 year ARM where we need to change interest rates from years 1 t
hrough 7
```

In [26]:
```
years
```

Out[26]:
```
[1991,
 1998,
 1999,
 2000,
 2001,
 2002,
 2003,
 2004,
 2005,
 2006,
 2007,
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 2017,
 2018,
 2019,
 2020]
```

In [33]:
```
interest_rates['year']= interest_rates.DATE.dt.year
interest_rates['month'] = interest_rates.DATE.dt.month
interest_rates['day'] = interest_rates.DATE.dt.day
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

In [44]:
```
annual_rates = interest_rates.loc[interest_rates[(interest_rates.year.isin
(years)) & (interest_rates.month == 1)].groupby(by=['year'])['day'].idxmin
()]
```

In [47]:
```
rates_to_consider = annual_rates[['year','MORTGAGE30US']]
```

In [61]:
```
 annual_rates_final = pd.concat([rates_to_consider, pd.DataFrame([[i, 9.56]
for i in range(1992,1998)], columns= ['year', 'MORTGAGE30US'])]).sort_value
s(by='year').reset_index(drop=True)
```

In [68]:
```
annual_rates_final.rename(columns = {'MORTGAGE30US':'rate'}, inplace=True)
```

In [83]:
```
def arm_payments(annual_rates_final, loan_amount):
    # create empty lists to store the results
    payments = [0]
    principal_pay_down = [0]
    interest_applied = [0]
    new_principal_balance = [loan_amount]

    # loop over the number of years
    for i in range(30*12):
            year = (i)//12
            rate = annual_rates_final.loc[year,'rate']/12/100
            term = 30*12-i
            fixed_payment = new_principal_balance[-1] * (rate * (1 + rate)
** term) / (((1 + rate) ** term) - 1)
            payments.append(fixed_payment)
            interest = rate * new_principal_balance[-1]
            interest_applied.append(interest)
            principal = fixed_payment - interest
            principal_pay_down.append(principal)
            new_principal_balance.append(new_principal_balance[-1]-principa
l)

    # create a new dataframe to store the results
    df = pd.DataFrame({'fixed_payment': payments,
                       'principal_pay_down': principal_pay_down,
                       'interest_applied': interest_applied,
                       'new_principal_balance': new_principal_balance})
    return df
```

In [76]:
```
loan_amount
```

Out[76]:
```
200000
```

In [84]:
```
arm_payments(annual_rates_final, loan_amount)
```

Out[84]:

	fixed_payment	principal_pay_down	interest_applied	new_principal_balance
0	0.000000	0.000000	0.000000	2.000000e+05
1	1690.470705	97.137372	1593.333333	1.999029e+05
2	1690.470705	97.911233	1592.559472	1.998050e+05
3	1690.470705	98.691259	1591.779446	1.997063e+05
4	1690.470705	99.477499	1590.993206	1.996068e+05
...	...	...	...	...
356	1163.953245	1146.078548	17.874698	4.619953e+03
357	1163.953245	1149.631391	14.321854	3.470322e+03
358	1163.953245	1153.195249	10.757997	2.317126e+03
359	1163.953245	1156.770154	7.183092	1.160356e+03
360	1163.953245	1160.356141	3.597104	3.842615e-11

361 rows × 4 columns