Importing all the neccesary Libraries

In [4]:
```
import pandas as pd
import numpy_financial as npf
from prettytable import PrettyTable,ALL
```

Defining all the functions required in the project

In [5]:
```
def mgmt_fee_calculation(mgmt_fee, Ending_or_bgn,bgn,gross_return):
    if Ending_or_bgn == "B":
        return  mgmt_fee * (bgn)
    else :
        return mgmt_fee * (1 + gross_return) * bgn
```

In [6]:
```
def catchup(bgn,gross_return,hurdle,incentive_rate):
 gross_value = ( 1 + gross_return) * bgn
 hurdle_NAV = ( 1+ hurdle ) * bgn
 Catchup_NAV = ((( hurdle ) * bgn)  / ( 1 - incentive_rate) ) + bgn

 if (hurdle_NAV < gross_value and gross_value <= Catchup_NAV ):
  return gross_value - hurdle_NAV
 elif gross_value > Catchup_NAV:
  return (gross_value - bgn )*( incentive_rate)
```

In [7]:
```
def incentive_fee_calculation(bgn,hurdle_rate,type,Gross_return,incentive_rate,HWM,mgmt_fee,net_or_independent):
  if net_or_independent == "I":
     Gross_return_abs = bgn * ( 1 + Gross_return)
  else :
    Gross_return_abs = bgn * ( 1 + Gross_return) - mgmt_fee

  if Gross_return_abs > HWM and Gross_return_abs > bgn * ( 1 + hurdle_rate):
    if type == "S":
        fees = ((Gross_return_abs - HWM ) * incentive_rate)
    elif type == "H":
        fees = (Gross_return_abs - (HWM * (1 + hurdle_rate))) * (incentive_rate)
    elif type == "C":
      fees = catchup(bgn,Gross_return,hurdle_rate,incentive_rate)
  else:
    fees = 0
  return fees
```

In [8]:
```
def clawback_provision(initial_amt, ending_AUM, incentive_rate, total_incentive_fee):
    total_profit = ending_AUM - initial_amt
    gp_entitlement = incentive_rate * max(0, total_profit)
    return max(0, total_incentive_fee - gp_entitlement)
```

In [9]:
```
def calculation_of_End_of_fund_value(bgn_balance, gross_return,total_fee):
    return (((bgn_balance *(1 + gross_return)-(total_fee))))
```

In [10]:
```
#inputs
hurdle_dic = {"S" : "Soft", "H":'Hard', "C":"Catchup"}
while True:
    print                          ("                                  Please Turn On CAPSLOCK")
    #initial_investment = int(input("Enter intial investment amount ----------------------------------------------------------------: "))
    print("""
    ---------------------This is a fund fee calculator-------------------------- \n
    The user is asked to provide inputs on mgmt fees , incentive structure and cashflows
    The Program uses these inputs to calculate to the FUnd fees of the end of the Investment Period
    and Provides a Brief summary of the fund's performance """)
    mgmt_fee =         float(input("Enter the management fee in decimals ( 1% as 0.01) ----------------------------------------: "))
    Calc_on=                 input("Management fee is calculated on Beginning(B) AUM or Ending (E) AUM - Enter 'B' or 'E' -------: ")
    hurdle =           float(input("Enter the hurdle rate ( 1% as 0.01) ------------------------------------------------------: "))
    h_type =                 input("Is it a soft(S) hurdle or a hard(H) hurdle or a Catchup clause - Enter S or H or C ----------: ")
    incentive_rate =   float(input("Enter the rate of incentive fee in decimals (1% as 0.01)-------------------------------------: "))
    calc_incentive = input("Incentive fee to be calculated net of mgmt fee(D) or Independent of Management Fee (I) --------------: ")
    Clawback_clause =       input( "Do we have a clawback provision in place ? Y/N ----------------------------------------------: ")

    # Create a PrettyTable to display inputs
    input_table = PrettyTable()
    input_table.field_names = ["Parameter", "Value"]
    input_table.align["Parameter"] = "l" # Left align the first column
    input_table.align["Value"] = "l"    # Left align the second column
    input_table.hrules = ALL # Horizontal rules for better readability

    #input_table.add_row(["Initial Investment", f"{initial_investment}"])
    input_table.add_row(["Management Fee", f"{mgmt_fee * 100}% ({'Bgn AUM' if Calc_on == 'B' else 'End AUM'})"])
    input_table.add_row(["Hurdle Rate", f"{hurdle * 100}% {hurdle_dic[h_type]} "]) #use dictionary for three
    input_table.add_row(["Incentive Fee Rate", f"{incentive_rate * 100}%  {"Net of Mgmt Fees" if calc_incentive == "D" else "Independent of Mgmt Fee)"}"])

    input_table.add_row(["Clawback Provision", f"{"Yes" if Clawback_clause == 'Y' else "No"}"])

    print("\n--- Summary of Initial Inputs ---")
    print(input_table)
    print("-----------------------------------")

    confirm = input("Are these inputs correct? (Y/N): ").strip().upper()
    if confirm == 'Y':
        break
    else:
        print("\nPlease re-enter the values.\n")

more_years = "Y"
```

```
                                Please Turn On CAPSLOCK

    ---------------------This is a fund fee calculator--------------------------

    The user is asked to provide inputs on mgmt fees , incentive structure and cashflows
    The Program uses these inputs to calculate to the FUnd fees of the end of the Investment Period
    and Provides a Brief summary of the fund's performance

--- Summary of Initial Inputs ---
+--------------------+---------------------------------+
| Parameter          | Value                           |
+--------------------+---------------------------------+
| Management Fee     | 1.0% (Bgn AUM)                  |
+--------------------+---------------------------------+
| Hurdle Rate        | 5.0% Soft                       |
+--------------------+---------------------------------+
| Incentive Fee Rate | 10.0%  Independent of Mgmt Fee) |
+--------------------+---------------------------------+
| Clawback Provision | No                              |
+--------------------+---------------------------------+
-----------------------------------
```

In [30]:

```python
Bgn_value = [0] #without any casflow change
cashflows = []
bgn_amt = [] # post cashflow change
gr = []
gross_AUM = []
management_fee = []
incentive_fee = []
end_value = []
high_watermark_history = [] # To store the HWM value for each period for display

more_years = "Y" # Ensure more_years is 'Y' to start the loop
print ('Enter the initial investment as the first cashflow in the box below')
while more_years == "Y":
    cf = float(input("Enter cashflow for this period (inflows to the fund + / outflows from fund - ): "))
    cashflows.append(cf)

    # Check for negative first cashflow, only applies to the very first cashflow entry.
    if len(cashflows) == 1 and cashflows[0] < 0:
        print ("First cashflow cannot be negative")
        cf =  int(input("Re-Enter a positive cashflow"))
        cashflows[0]= cf

    # Calculate beginning AUM for the current period (previous End AUM + current cashflow)
    bgn_bal_current_period = Bgn_value[-1] + cf
    bgn_amt.append(bgn_bal_current_period)

    # Set the initial high_watermark for the first period
    if len(bgn_amt) == 1: # This is the first period
        current_high_watermark = bgn_bal_current_period # HWM starts with the initial AUM for the first period
```

```
#Modify HWM incase of any fund inflows and outflows
if len(bgn_amt) != 1 and cf != 0:
   current_high_watermark = high_watermark_history[-1] * (bgn_bal_current_period/Bgn_value[-1])

# Store the HWM that applies to this period's incentive fee calculation
high_watermark_for_this_period = current_high_watermark
high_watermark_history.append(high_watermark_for_this_period) # To display in the table

gross_return = float(input("Enter gross return in decimals : "))
gr.append(gross_return)

gross_AUM_current_period = bgn_amt[-1] * (1 + gross_return)
gross_AUM.append(gross_AUM_current_period) # Append gross AUM to its list

# Calculate individual fees first
current_mgmt_fee = mgmt_fee_calculation(mgmt_fee, Calc_on, bgn_amt[-1], gross_return)

# Incentive fee is calculated only if Gross AUM exceeds the HWM for this period
if gross_AUM_current_period > high_watermark_for_this_period:
  current_incentive_fee = incentive_fee_calculation(bgn_amt[-1], hurdle, h_type, gross_return, incentive_rate, high_watermark_for_this_period, current_mgmt_fe
else:
  current_incentive_fee = 0 # If not exceeding HWM, incentive fee is 0

total_fees = current_mgmt_fee + current_incentive_fee

# Calculate end of fund value
fund_end_value = calculation_of_End_of_fund_value(bgn_amt[-1], gross_return, total_fees)

# Update HWM for the *next* period: HWM is the maximum of its current value and the fund's end value (after fees).
current_high_watermark = max(current_high_watermark, fund_end_value)

# Update lists with calculated values
end_value.append(round(fund_end_value,2))
management_fee.append(round(current_mgmt_fee,2))
incentive_fee.append(round(current_incentive_fee,2))

# The next beginning value is the current fund_end_value
Bgn_value.append(round(fund_end_value,2))

num_periods = len(management_fee)

data = {
  'Period': list(range(1, num_periods + 1)),
  'Beginning AUM': Bgn_value[:-1],
  'Cashflow for the Period': cashflows,
  'BGN AUM + Cashflow' : bgn_amt,
  'Gross Return ' : [f"{a:.0%}"for a in gr],
  'Gross AUM ' : gross_AUM,
  "High Water Mark" : high_watermark_history, # Display the HWM used for each period
  'Management Fee': management_fee,
```

```
      'Incentive Fee': incentive_fee,
      'End AUM': end_value
    }

    df = pd.DataFrame(data)
    print(df.to_string(index=False))
    print ("\n" *3)
    more_years = ""
    while more_years.upper() not in ("Y" , "N"):
        more_years = (input("Do you have more years left ? Y/N : ")).upper()
```

```
Enter the initial investment as the first cashflow in the box below
 Period  Beginning AUM  Cashflow for the Period  BGN AUM + Cashflow Gross Return   Gross AUM   High Water Mark  Management Fee  Incentive Fee  End AUM
      1              0                    100.0               100.0           20%       120.0            100.0             1.0            2.0    117.0



 Period  Beginning AUM  Cashflow for the Period  BGN AUM + Cashflow Gross Return   Gross AUM   High Water Mark  Management Fee  Incentive Fee  End AUM
      1            0.0                    100.0               100.0           20%       120.0            100.0            1.00           2.00   117.00
      2          117.0                      0.0               117.0           10%       128.7            117.0            1.17           1.17   126.36



 Period  Beginning AUM  Cashflow for the Period  BGN AUM + Cashflow Gross Return   Gross AUM   High Water Mark  Management Fee  Incentive Fee  End AUM
      1           0.00                    100.0              100.00           20%     120.000       100.000000            1.00           2.00   117.00
      2         117.00                      0.0              117.00           10%     128.700       117.000000            1.17           1.17   126.36
      3         126.36                    -20.0              106.36          -10%      95.724        98.481481            1.06           0.00    94.66
```

In [ ]:

```
invested_cf = sum(cashflows)
cashflows.append(-(end_value[-1]))
irr = f"{round(npf.irr(cashflows),2)*100}%"


performance_table = PrettyTable()
performance_table.field_names = ["Parameter", "Value"]
performance_table.add_row(["Total Invested Amount ", round(invested_cf)])
performance_table.add_row(["Total Managment Fees Paid ", round(sum(management_fee))])
performance_table.add_row(["Total Incentive Fees Paid", round(sum(incentive_fee))])
performance_table.add_row(["Closing Value of Investment", end_value[-1]])
performance_table.add_row(["IRR earned on the investment ", irr ])
performance_table.add_row(["Do we have a  Clawback Provision Apply ? ", f"{"Yes" if Clawback_clause == 'Y' else "No"}"])
if Clawback_clause == 'Y':
  performance_table.add_row(["Amount to be clawed back ", round(clawback_provision(bgn_amt[0],end_value[-1],incentive_rate,sum(incentive_fee)),2)])
print(performance_table)
```

| Parameter | Value |
|---|---|
| Total Invested Amount | -15 |
| Total Managment Fees Paid | 3 |
| Total Incentive Fees Paid | 3 |
| Closing Value of Investment | 94.66 |
| IRR earned on the investment | 25.0% |
| Do we have a Clawback Provision Apply ? | No |