In [176]:
```
#This is a cell for imports.
#It must be run before any of the following cells
import datetime
import time
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [160]:
```
#This is a function that extracts and prints the current time in the ISO format.
def getDate():
    t = time.localtime(time.time())
    print("Today's ISO format is:", end="\t")
    print(str(t.tm_year)+"-"+str(t.tm_mon)+"-"+str(t.tm_mday)+"W"+str(datetime.date(t.tm_year, t.tm_mon,t.tm_mday).isocalendar
().week)+"T"+str(t.tm_hour)+":"+str(t.tm_min)+":"+str(t.tm_sec))

getDate()
```

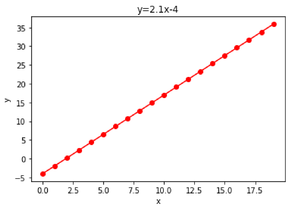
```
Today's ISO format is:  2022-3-11W10T22:22:26
```

In [8]:
```
#This is a function that allows us to print pyramids of any length, the times we want.
#@param limit is the height of the pyramid
#@param numer is the number of pyramids we are going to print
def forCNLoops(limit, number):
    j = 0
    for l in range(0,number*2):
        if j >= 5:
            for k in range(0, limit):
                print('cn '*j)
                j=j-1
        else:
            for i in range(0, limit):
                print('cn '*j)
                j = j+1
forCNLoops(5, 1)
```

```
cn
cn cn
cn cn cn
cn cn cn cn
cn cn cn cn cn
cn cn cn cn
cn cn cn
cn cn
cn
```

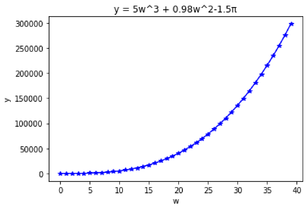
In [177]:
```
#This function defines the constraints of the red plot function
def redPlot():
    y1=[]
    for i in x:
        y1.append(2.1*i-4)
    return y1

#Code that shows and entitles the red plot
x=list(range(0, 20))
y1=redPlot()
plt.plot(x,y1, marker="o", color="red")
plt.xlabel('x')
plt.ylabel('y')
plt.title('y=2.1x-4')
```

Out[177]:
```
Text(0.5, 1.0, 'y=2.1x-4')
```

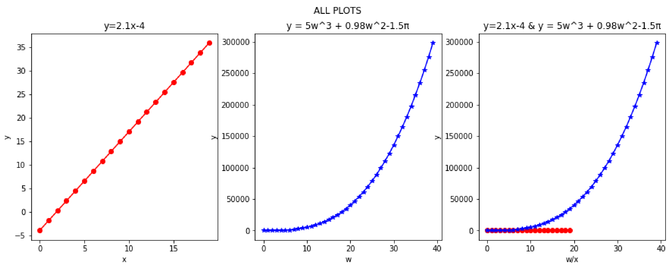


In [104]:
```
#This function defines the constraints of the blue plot function
def bluePlot():
    y2=[]
    pi=3.14159
    for i in w:
        y2.append(5*i**3+0.98*i**2-1.5*pi)
    return y2
#Code that shows and entitles the blue plot
w=list(range(0, 40))
y2=bluePlot()
plt.plot(w,y2, marker="*", color="blue")
plt.xlabel('w')
plt.ylabel('y')
plt.title('y = 5w^3 + 0.98w^2-1.5π')
```

Out[104]:
```
Text(0.5, 1.0, 'y = 5w^3 + 0.98w^2-1.5π')
```



In [174]:
```
#This function plot the previous blue and red plots and also merges them into a third comparison plot.
def multiPlots():
    plt.figure(figsize=(15, 5))
    plt.suptitle('ALL PLOTS')
    plt.subplot(131)
    y1=redPlot()
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('y=2.1x-4')
    plt.plot(x,y1,marker="o", color="red")
    plt.subplot(132)
    plt.xlabel('w')
    plt.ylabel('y')
    plt.title('y = 5w^3 + 0.98w^2-1.5π')
    y2=bluePlot()
    plt.plot(w,y2,marker="*", color="blue")
    plt.subplot(133)
    plt.xlabel('w/x')
    plt.ylabel('y')
    plt.title('y=2.1x-4 & y = 5w^3 + 0.98w^2-1.5π')
    plt.plot(x,y1,marker="o", color="red")
    plt.plot(w,y2,marker="*", color="blue")
    plt.savefig("myPlotsPDF.pdf", format="pdf", bbox_inches="tight")
    plt.show()

multiPlots()
```



In [183]:
```
#This function allows us to print a pythagorean table for every size given in the parameters.
#@param x, y are the number of columns and rows of the table respectively
def pythagoreanTable(x, y):
    A = []
    B = []
    print("", end="\t")
    for i in range(0, y+1):
        A.append(i)
        for j in range(0, x+1):
            B.append(j)
            if A[i]*B[j]!=0:
                print(str(A[i]*B[j]), end="\t")
            elif B[j]<=x and B[j]!=0:
                print(str(B[j]), end="\t")
            elif A[i]!=0:
                print(str(A[i]), end="\t")
        print()


pythagoreanTable(10,10)
```

```
        1       2       3       4       5       6       7       8       9       10
1       1       2       3       4       5       6       7       8       9       10
2       2       4       6       8       10      12      14      16      18      20
3       3       6       9       12      15      18      21      24      27      30
4       4       8       12      16      20      24      28      32      36      40
5       5       10      15      20      25      30      35      40      45      50
6       6       12      18      24      30      36      42      48      54      60
7       7       14      21      28      35      42      49      56      63      70
8       8       16      24      32      40      48      56      64      72      80
9       9       18      27      36      45      54      63      72      81      90
10      10      20      30      40      50      60      70      80      90      100
```