Lab4

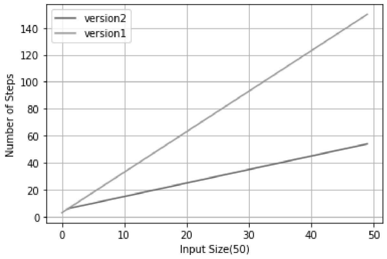
Data Structures and Algorithms

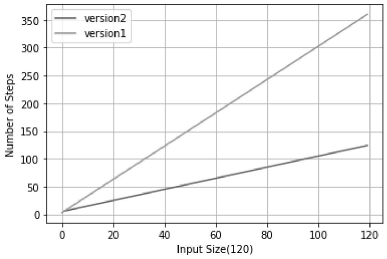
Maira usman

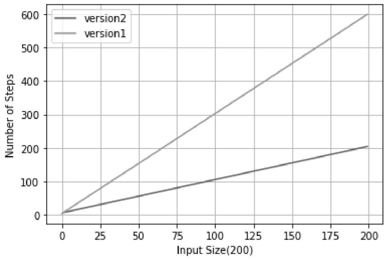
21B-011-SE

In [8]:

```
from matplotlib import pyplot as plt
import numpy as np
def version1(n):
    totalSum = 0
    matrix= np.random.randint(10, size=(n, n))
    rowSum=[0]*n
    counter=3
    for i in range(0,n):
        rowSum[i] = 0
        counter+=1
        for j in range(0, n ) :
            rowSum[i] = rowSum[i] + matrix[i,j]
        totalSum = totalSum + matrix[i,j]
        counter+=2
    return counter
def version2(n):
    matrix= np.random.randint(10, size=(n, n))
    totalSum = 0 # Version 1 18 matrix= np.random.randint(10, size=(n, n))
    rowSum=[0]*n
    counter=3 #Counts the number of statement excuted , excluding the counter updates
    for i in range(0,n,1):
        rowSum[i] = 0
        counter+=1
        for j in range(0, n ) :
            rowSum[i] = rowSum[i] + matrix[i,j]
            counter+=1
        totalSum = totalSum + rowSum[i]
        counter+=1
        return counter
def simulation(n):
    steps_version1=[0]*n
    steps_version2 = [0] * n
    for i in range(0,n):
        steps_version1[i]=version1(i)
        steps_version2[i]=version2(i)
    x=list(range(n))
    plt.plot(x,steps_version2)
    plt.plot(x, steps_version1)
    plt.grid(which='both')
    plt.xlabel(f'Input Size({n})')
    plt.ylabel('Number of Steps')
    plt.legend(['version2','version1'])
    plt.show()
simulation(50)
simulation(120)
simulation(200)
```



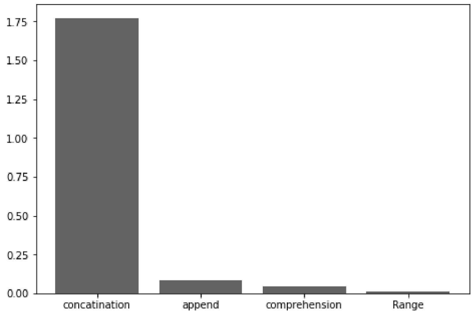




In [1]:

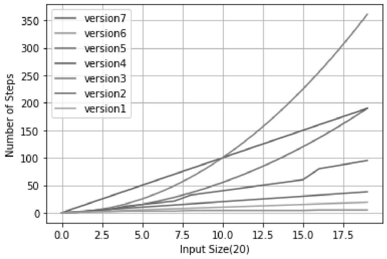
```
from timeit import Timer
import matplotlib.pyplot as plt
def concatenation():
    l = []
    for i in range(1000):
        l = l + [i]
def append():
    l = []
    for i in range(1000):
        l.append(i)
def comprehension():
    l = [i for i in range(1000)]
def rangeFunction():
    l = list(range(1000))
t1 = Timer("concatenation()", "from __main__ import concatenation")
concatTime = t1.timeit(number=1000)
print("concatination ", concatTime , "milliseconds")
t2 = Timer("append()", "from __main__ import append")
appendTime = t2.timeit(number=1000)
print("append ", appendTime , "milliseconds")
t3 = Timer("comprehension()", "from __main__ import comprehension")
compTime= t3.timeit(number=1000)
print("comprehension ", compTime , "milliseconds")
t4 = Timer("rangeFunction()", "from __main__ import rangeFunction")
rangeTime = t4.timeit(number=1000)
print("list range ",rangeTime , "milliseconds")

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['concatination', 'append', 'comprehension', 'Range']
students = [concatTime ,appendTime ,compTime ,rangeTime]
ax.bar(langs,students)
plt.show()
```

```
concatination  1.7751855 milliseconds
append  0.08531140000000015 milliseconds
comprehension  0.04348749999999946 milliseconds
list range  0.01481530000000042 milliseconds
```



In [27]:

```
def ex1(n):
    count=0
    for i in range(n):
        count+=1
    return count
def ex2(n):
    count=0
    for i in range(n):
        count+=1
    for j in range(n):
        count+=1
    return count
def ex3(n):
    count=0
    for i in range(n):
        for j in range(n):
            count+=1
    return count
def ex4(n):
    count=0
    for i in range(n):
        for j in range(10):
            count+=1
    return count
def ex5(n):
    count=0
    for i in range(n):
        for j in range(i+1):
            count+=1
    return count
def ex6( n ):
    count = 0
    i = n
    while i >= 1 :
        count += 1
        i = i // 2
    return count
def ex7(n):
    count=0
    for i in range(n):
        count+=ex6(n)
    return count
def simulation(n):
    steps_version1=[0]*n
    steps_version2 = [0] * n
    steps_version3 = [0] * n
    steps_version4 = [0] * n
    steps_version5 = [0] * n
    steps_version6 = [0] * n
    steps_version7 = [0] * n
    for i in range(0,n):
        steps_version1[i]=ex1(i)
        steps_version2[i]=ex2(i)
        steps_version3[i]=ex3(i)
        steps_version4[i]=ex4(i)
        steps_version5[i]=ex5(i)
        steps_version6[i]=ex6(i)
        steps_version7[i]=ex7(i)
    x=list(range(n))
    plt.plot(x,steps_version7)
    plt.plot(x,steps_version6)
    plt.plot(x,steps_version5)
    plt.plot(x,steps_version4)
    plt.plot(x,steps_version3)
    plt.plot(x,steps_version2)
    plt.plot(x,steps_version1)
    plt.grid(which='both')
    plt.xlabel(f'Input Size({n})')
    plt.ylabel('Number of Steps')
    plt.legend(['version7','version6','version5','version4','version3','version2','version1'])
    plt.show()
simulation(20)
```



In [23]:

```
import random
count=0
lst=[i for i in range(1000)]

print("shuffledlist element 50 : ", lst.index(50))
case=[]
for i in range(50):
    random.shuffle(lst)
    case.append(lst.index(50))
print("best case: ",min(case))
print("worst case: ",min(case))
print("average case: ",sum(case)//len(case))
```

```
shuffledlist element 50 :  50
best case:  24
worst case:  24
average case:  468
```

In [ ]: