Task 15

Registration ID : SIRSS1038

Leela Satya Kartheek Raja

Q1) Given an array Arr of N positive integers, find K largest elements from the array. The output elements should be printed in decreasing order.

In [6]:

```
def KL(arr,k):
    arr.sort(reverse=True)

    for i in range(k):
        print(arr[i],end=" ")
```

In [7]:

```
# arr=[12,5,787,1,23]
# k=2
arr=[1,23,12,9,30,2,50]
k=3
KL(arr,k)
```

```
50 30 23
```

Q2) You are given weights and values of N items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Note that we have only one quantity of each item. In other words, given two integer arrays val[0..N-1] and wt[0..N-1] which represent values and weights associated with N items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W.

You cannot break an item, either pick the complete item or don't pick it (0-1 property).

In [8]:

```
def KS(w,wt,val,n):
    if n==0 or w==0:
        return 0

    if (wt[n-1]>w):
        return KS(w,wt,val,n-1)

    else:
        return max(val[n-1]+KS(w-wt[n-1],wt,val,n-1),KS(w,wt,val,n-1))
```

In [9]:

```
# val=[1,2,3]
# wt=[4,5,1]
# w=4

val=[1,2,3]
wt=[4,5,6]
w=3

n=len(val)
```

```
print(KS(w,wt,val,n))
```

```
0
```

Q3) Given two sequences, find the length of longest subsequence present in both of them. Both the strings are of uppercase.

In [10]:

```
def LCSLength(X, Y, m, n):
    if m == 0 or n == 0:
        return 0

    if X[m - 1] == Y[n - 1]:
        return LCSLength(X, Y, m - 1, n - 1) + 1

    return max(LCSLength(X, Y, m, n - 1), LCSLength(X, Y, m - 1, n))


if __name__ == '__main__':

    X="ABCDGH"
    Y="AEDFHR"



    print("Length is ",LCSLength(X, Y, len(X), len(Y)))
```

```
Length is  3
```