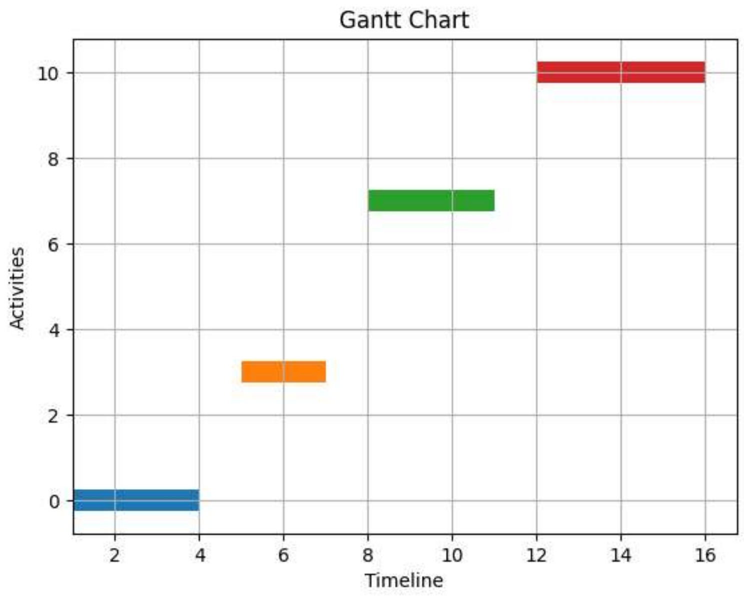
```
### PART (A)
### Aim: Implement activity selection algorithm for the given scenario.
### Problem Definition: In the single-machine scheduling problem, we are given a set of n activities Ai.
###  Each job i has a starting time si, deadline di and profit pi. At any time instant, we can do only one job.
### Doing a job i earns a profit pi. Generate a solution to select the largest set of mutually compatible jobs and
### calculate the total profit generated by the machine. The greedy algorithm for single-machine scheduling selects the job using activity se
```

```
import matplotlib.pyplot as plt
import numpy as np
index=[0]
def ActivitySelection(si,di,profit):
    n=len(si)
    k=0
    p=10
    A=['a1']
    for i in range(1,n):
        if(si[i]>=di[k]):
            index.append(i)
            A.append(f"a{i+1}")
            k=i
            p+=profit[k]
    print("Activity Selected : ",A)
    print("Total Profit : ",p)
    print("Index : ",index)

si=[1,3,0,5,3,5,6,8,8,2,12]
di=[4,5,6,7,9,9,10,11,12,14,16]
profit=[10,15,14,12,20,30,32,28,30,40,45]
ActivitySelection(si,di,profit)
fig2,ax2=plt.subplots()
for i in index:
    print(si[i],di[i])
    start_time=si[i]
    end_time=di[i]
    ax2.barh(i,end_time-start_time,left=start_time,height=0.5,align='center')
ax2.set_xlabel('Timeline')
ax2.set_ylabel('Activities')
ax2.set_title('Gantt Chart')
ax2.grid(True)
plt.show()
```

```
Activity Selected :  ['a1', 'a4', 'a8', 'a11']
Total Profit :  95
Index :  [0, 3, 7, 10]
1 4
5 7
8 11
12 16
```



```
### PART (B)
### Aim: Given three stacks of the positive numbers,
### the task is to find the possible equal maximum sum of the stacks with the removal of top elements allowed.


def MSP(stacks):
    sums=[sum(stacks[0]), sum(stacks[1]), sum(stacks[2])]
    while(sums[0] != sums[1] != sums[2]):
        maxm=max(sums)
        index=sums.index(maxm)
        stacks[index].pop()
        sums=[sum(stacks[0]), sum(stacks[1]), sum(stacks[2])]
    return sums[0]



stk1=[3,5,8,5]
stk2=[2,2,4,9,6,5]
stk3=[2,1,2,3,5]
stack=[stk1,stk2,stk3]
stack
maximum_sum=MSP(stack)
maximum_sum
```

```
8
```