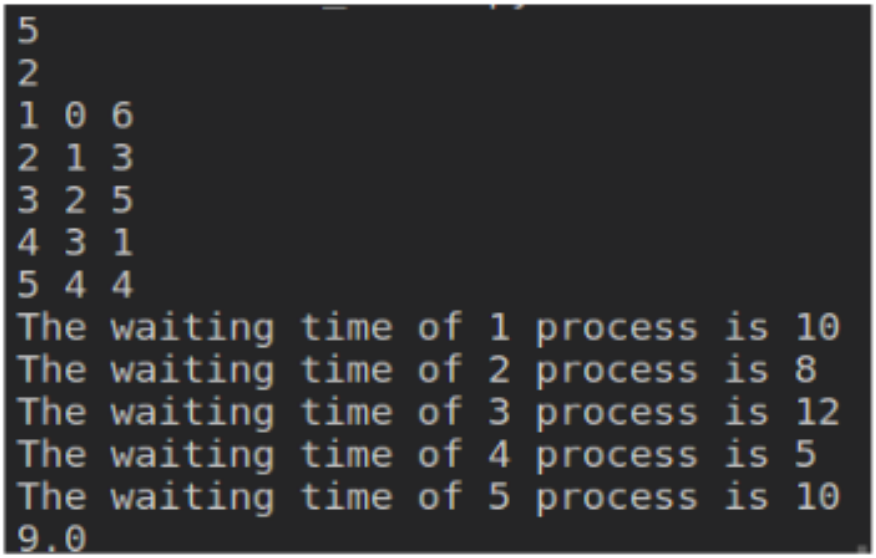
Round Robin:

```
operation, readyQueue, executionQueue, isExecution = [], [], [], []
completionTime, waitingTime, turnaroundTime, burstTimeList = [], [], [], []
noOfProcess = int(input())
timeQuantum = int(input())
totalTime = 0
for _ in range(noOfProcess):
  processes, arrivalTime, burstTime = map(int, input().strip().split())
totalTime += burstTime
operation.append((arrivalTime, processes, burstTime))
burstTimeList.append(burstTime)
isExecution.append(-1)
completionTime.append(-1)
waitingTime.append(-1)
turnaroundTime.append(-1)
operation.sort()
readyQueue.append(operation[0])
isExecution[operation[0][1]-1] = 1
sum = 0
while (sum != totalTime):
  x = readyQueue[0]
readyQueue.pop(0)
if (x[2] < timeQuantum):
  sum += x[2]
first_parameter = x[0]
second_parameter = x[1]
third_parameter = 0
else:
  sum += timeQuantum
first_parameter = x[0]
second_parameter = x[1]
third_parameter = x[2]-timeQuantum
operation.remove(x)
joiningList = (first_parameter, second_parameter, third_parameter)
operation.append(joiningList)operation.sort()
executionQueue.append((second_parameter, sum))
for i in range(len(operation)):
  if (operation[i][0] <= sum and isExecution[i] == -1):
    readyQueue.append(operation[i])
isExecution[operation[i][1]-1] = 1
if (third_parameter != 0):
  readyQueue.append(joiningList)
count = 0
```

```
for i in range(len(executionQueue)-1, -1, -1):
  if (completionTime[executionQueue[i][0]-1] == -1):
    completionTime[executionQueue[i][0]-1] = executionQueue[i][1]
    count += 1
  elif (count == 5):
    break
for i in range(noOfProcess):
  turnaroundTime[i] = completionTime[i]-operation[i][0]
avg = 0
for i in range(noOfProcess):
  waitingTime[i] = turnaroundTime[i]-burstTimeList[i]
avg += waitingTime[i]
for i in range(noOfProcess):
  print(f'The waiting time of {i+1} process is {waitingTime[i]}')
print(avg/noOfProcess)
```

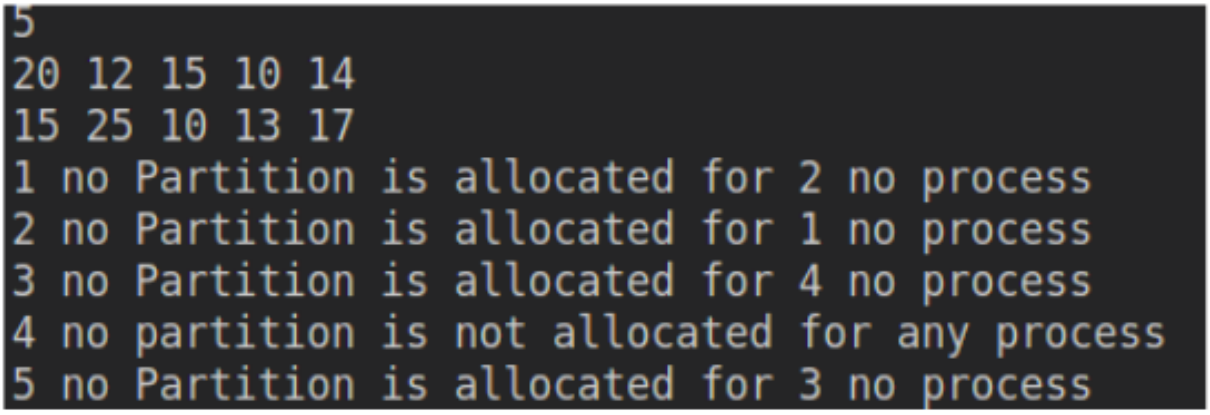
Input and Output



First Fit:

```
process,memory,allocation = [],[],[]
noOfProcess=int(input())
#take input
for _ in range(noOfProcess):
  allocation.append(-1)
process=list(map(int, input().strip().split()))[:noOfProcess]
memory=list(map(int, input().strip().split()))[:noOfProcess]
for i in range(noOfProcess):
  processSize= process[i]
for j in range(noOfProcess):
  if(processSize<=memory[j] and allocation[j]==-1):
     allocation[j]=i+1
  break
for k in range(noOfProcess):
  if(allocation[k]==-1):
     print(f'{k+1} no partition is not allocated for any process')
  else:
     print(f'{k+1} no Partition is allocated for {allocation[k]} no process')
```

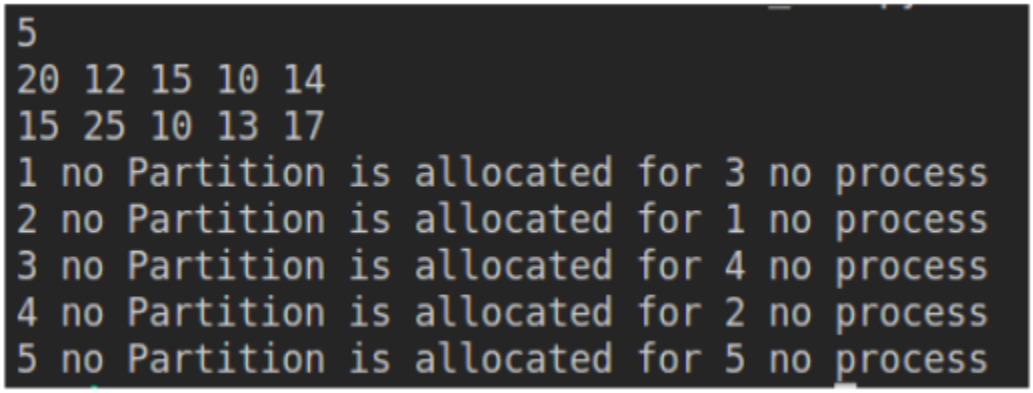
Input and Output



Best fit

```
process,memory,allocation = [],[],[]
noOfProcess=int(input())
#take input
for _ in range(noOfProcess):
        allocation.append(-1)
        process=list(map(int, input().strip().split()))[:noOfProcess]
        memory=list(map(int, input().strip().split()))[:noOfProcess]
for i in range(noOfProcess):
        processSize= process[i] #20
matching=[]
for j in range(noOfProcess):
        if(processSize<=memory[j] and allocation[j]==-1):
                matching.append((memory[j]-processSize,j))
                matching.sort()
                allocation[matching[0][1]]=i+1
for k in range(noOfProcess):
        if (allocation[k] == -1):
        print(f'{k+1} no partition is not allocated for any process')
else:
        print(f'{k+1} no Partition is allocated for {allocation[k]} no process')
```

Input and Output



Worst fit

```
process, memory, allocation = [], [], []
noOfProcess = int(input())
#take input
for _ in range(noOfProcess):
        allocation.append(-1)
        process = list(map(int, input().strip().split()))[:noOfProcess]
        memory = list(map(int, input().strip().split()))[:noOfProcess]
for i in range(noOfProcess):
        processSize = process[i]
        matching = []
for j in range(noOfProcess):
        if (processSize <= memory[j] and allocation[j] == -1):
                matching.append((memory[j]-processSize, j))
                matching.sort(reverse=True)
        if(len(matching)>0):
                allocation[matching[0][1]] = i+1
for k in range(noOfProcess):
        if (allocation[k] == -1):
                print(f'{k+1} no partition is not allocated for any process')
        else:
                print(f'{k+1} no Partition is allocated for {allocation[k]} no process')
```

Input and Output