**EXPERIMENT NO. 3**

**AIM:** Write a program to implement Round Robin scheduling algorithm for process management.

**RESOURCES REQUIRED:**

H/W Requirements: P-IV and above, Ram 128 MB, Printer, Internet Connection.

S/W Requirements: Python compiler.

**THEORY:**

Round Robin is a [CPU scheduling algorithm](http://quiz.geeksforgeeks.org/gate-notes-operating-system-process-scheduling/) where each process is assigned a fixed time slot in a cyclic way.

It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.

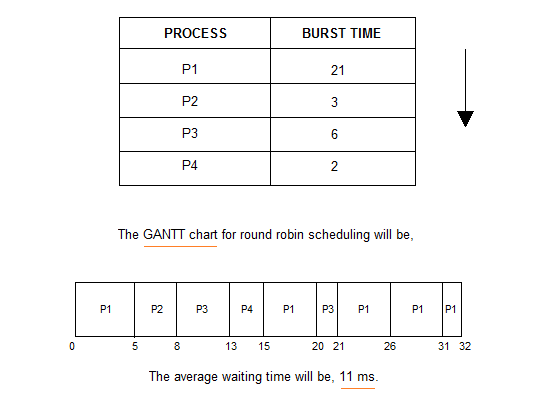
One of the most commonly used technique in CPU scheduling as a core.

It is pre-emptive as processes are assigned CPU only for a fixed slice of time at most.

The disadvantage of it is more overhead of context switching.

**Steps to find waiting times of all processes:**

1. Create an array **rem\_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array).
2. Create another array **wt[]** to store waiting times of process. Initialize this array as 0.
3. Initialize time: t = 0.
4. Keep traversing the all processes while all processes are not done. Do following for i’th process if it is not done yet.
5. If rem\_bt[i] > quantum
6. t = t +quantum
7. rem\_bt[i] -= quantum
8. Else
9. t = t + rem\_bt[i]
10. wt[i] = t – bt[i]
11. rem\_bt[i] = 0; // This process is over.



**CONCLUSION:** Hence, we have implemented a program on Round Robin scheduling algorithm on process management.

**CODE:**

from prettytable import PrettyTable

def rr():

pid,at,bt,tt,wt = [],[],[],[],[]

print()

z = int(input("Enter number of Process: "))

ct = [0]\*z

quantum = int(input("Enter the quantum time: "))

print()

for i in range(0,z):

pid.append(int(input("Enter Proccess id:")))

print()

at.append(int(input("Enter arrival time:")))

print()

bt.append(int(input("Enter burst time:")))

print()

for i in range(0,z):

min = [pid[i],at[i],bt[i]]

j = i-1

while(j>=0 and at[j]>min[1]):

at[j+1],pid[j+1],bt[j+1] = at[j],pid[j],bt[j]

j = j-1

pid[j+1],at[j+1],bt[j+1] = min[0],min[1],min[2]

rem\_bt = bt.copy()

tot = 0

while(True):

status = True

for x in range(0,z):

if rem\_bt[x] > 0:

status = False

if (rem\_bt[x]-quantum)>0:

rem\_bt[x] -= quantum

tot += quantum

else:

tot += rem\_bt[x]

ct[x] = tot+1

rem\_bt[x] = 0

if status:

break

tt.append(ct[0]-at[0])

wt.append(tt[0]-bt[0])

for i in range(1,z):

tt.append(ct[i]-at[i])

wt.append(tt[i]-bt[i])

x = PrettyTable()

x.field\_names = ["Process id","Arrival Time","Burst Time","Completion Time","Turnaround Time","Waiting Time"]

for a,b,c,d,e,f in zip(pid,at,bt,ct,tt,wt):

x.add\_row([a,b,c,d,e,f])

print(x)

print("Total turnaround time: "+str(sum(tt))+"\nTotal waiting time: "+str(sum(wt)))

print("Average turnaround time: "+str(sum(tt)/z)+"\nAverage waiting time: "+str(sum(wt)/z))

if \_\_name\_\_ == "\_\_main\_\_":

print("55\_Adnan\_Shaikh")

rr()

**OUTPUT:**

