NAME: ALISTAIR SALDANHA

SAPID: 60009200024

BATCH: K1

EXPERIMENT-5

In [16]:

```python
from prettytable import PrettyTable
x = PrettyTable()
```

In [17]:

```python
# processes = [['P1', 0, 7], ['P2', 0, 5], ['P3', 0, 3], ['P4', 0, 1], ['P5',0, 2], ['P6'
, 0, 1]]
processes = [('P1', [0, 7]), ('P2', [0, 5]), ('P3', [0, 3]), ('P4', [0, 1]), ('P5', [0,
2]), ('P6', [0, 1])]
processes1 = processes.copy()
```

In [18]:

```python
def gantt_c(processes1):
    counter = 0
    ready_queue = []
    gantt_chart = []
    burst_time = 0
    time=0
    completion_time = {}

    for i in range(len(processes1)):
        burst_time += processes1[i][1][1]

    gantt_chart.append(0)
    while(counter <= burst_time):
        for i in range(len(processes1)):
            arrival_time = processes1[i][1][0]
            if((counter == arrival_time) and (processes1[i] not in ready_queue)):
                ready_queue.append(processes1[i])
        if(len(ready_queue) > 1):
            ready_queue = sorted(ready_queue, key=lambda x: x[1][1])
        gantt_chart.append(ready_queue[0][0])
        time += 1
        gantt_chart.append(time)
        if(ready_queue[0][1][1] > 1):
            ready_queue[0][1][1] -= 1
        else:
            completion_time[ready_queue[0][0]] = time
            ready_queue.pop(0)
        if(ready_queue == []):
            break
        counter += 1
    return gantt_chart, completion_time

gantt_chart, completion_time = gantt_c(processes1)
completion_time = sorted(completion_time.items(), key=lambda x: x[0])

def display_gc(gantt_chart):
    for i in range(len(gantt_chart)):
        print(f"{gantt_chart[i]}", end="")
        if(i < len(gantt_chart)-1):
            print("-->", end="")
```

```
display_gc(gantt_chart)
```

```
0-->P4-->1-->P6-->2-->P5-->3-->P5-->4-->P3-->5-->P3-->6-->P3-->7-->P2-->8-->P2-->9-->P2--
>10-->P2-->11-->P2-->12-->P1-->13-->P1-->14-->P1-->15-->P1-->16-->P1-->17-->P1-->18-->P1-
->19
```

In [19]:

```
processes = [('P1', [0, 7]), ('P2', [0, 5]), ('P3', [0, 3]), ('P4', [0, 1]), ('P5', [0,
2]), ('P6', [0, 1])]
def calc_TAT(completion_time,processes):
    TAT = []
    for i in range(len(processes)):
        TAT.append(completion_time[i][1] - processes[i][1][0])
    return TAT
TAT = calc_TAT(completion_time,processes)
print("TAT :",TAT)

def calc_WT(TAT,processes):
    WT = []
    for i in range(len(processes)):
        WT.append(TAT[i] - processes[i][1][1])
    return WT
WT = calc_WT(TAT,processes)
print("WT :",WT)

def avg_WT(WT):
    avg_WT = 0.0
    for i in range(len(WT)):
        avg_WT += WT[i]
    avg_WT = (avg_WT/len(WT))
    return round(avg_WT,3)
average_WT = avg_WT(WT)

def avg_TAT(TAT):
    avg_TAT = 0.0
    for i in range(len(TAT)):
        avg_TAT += TAT[i]
    avg_TAT = (avg_TAT/len(TAT))
    return round(avg_TAT,3)
average_TAT = avg_TAT(TAT)
```

```
TAT : [19, 12, 7, 1, 4, 2]
WT : [12, 7, 4, 0, 2, 1]
```

In [20]:

```
def table(processes):
    x.field_names = ["Process","AT","BT","FT","TAT","WT"]
    for i in range(len(processes)):
        x.add_row([f"P{i+1}",processes[i][1][0],processes[i][1][1], completion_time[i][1
],TAT[i],WT[i]])
    print(x)

table(processes)
print(f"Average Waiting Time: {average_WT}")
```

```
+---------+----+----+----+-----+----+
| Process | AT | BT | FT | TAT | WT |
+---------+----+----+----+-----+----+
|    P1   | 0  | 7  | 19 |  19 | 12 |
|    P2   | 0  | 5  | 12 |  12 | 7  |
|    P3   | 0  | 3  | 7  |  7  | 4  |
|    P4   | 0  | 1  | 1  |  1  | 0  |
|    P5   | 0  | 2  | 4  |  4  | 2  |
|    P6   | 0  | 1  | 2  |  2  | 1  |
+---------+----+----+----+-----+----+
Average Waiting Time: 4.333
```

In [21]:

```
def round_robin(process, TS):
    arrival = process[0]
    burst = process[1]
    n = len(burst)

    # Sort wrt Arrival
    for i in range(len(arrival)):
        for j in range(len(arrival)-i-1):
            if arrival[j] > arrival[j+1]:
                arrival[j], arrival[j+1] = arrival[j+1], arrival[j]
                burst[j], burst[j+1] = burst[j+1], burst[j]

    burst_copy = burst.copy()
    # Gantt chart and Process
    burst_sum = [0]
    max_time = 0
    for i in range(len(burst)):
        max_time += burst[i]

    x = 0
    val = 0
    while(1):
      if(x>2):
        x = 0
      if(burst_copy[x]<TS):
        val += burst_copy[x]
        burst_copy[x] = 0
        burst_sum.append(val)
      else:
        val += TS
        burst_copy[x] = burst_copy[x] - 2
        burst_sum.append(val)
      x += 1
      res = all(ele == 0 for ele in burst_copy)
      if(res):
        break
    burst_sum = list(set(burst_sum))
    print(f'Gantt Chart: {burst_sum}\n')

    wt = [0] * n
    tat = [0] * n
    rem_bt = [0] * n

    # Copy the burst time into rt[]
    for i in range(n):
        rem_bt[i] = burst[i]
    t = 0 # Current time

    while(1):
      done = True
      for i in range(n):
        if (rem_bt[i] > 0) :
          done = False # There is a pending process

          if (rem_bt[i] > TS) :
              t += TS
              rem_bt[i] -= TS
          else:
              t = t + rem_bt[i]
              wt[i] = t - burst[i]
              rem_bt[i] = 0

      # If all processes are done
      if (done == True):
          break

    for i in range(n):
        tat[i] = burst[i] + wt[i]

    print("Processes    Burst Time     Waiting",
                     "Time    Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):

        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", i + 1, "\t\t", burst[i],
              "\t\t", wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.2f "%(total_wt -sum(arrival) /n) )
    print("Average turn around time = %.2f "% (total_tat - sum(arrival) / n))
```

In [22]:

```
process = [[0,0,0], [9,4,9]]
TS = 3
print('Round Robin Algorithm')
round_robin(process, TS)
```

```
Round Robin Algorithm
Gantt Chart: [0, 3, 6, 9, 12, 14, 17, 20, 23, 26, 29, 30, 31]

Processes    Burst Time     Waiting Time    Turn-Around Time
  1    9    10     19
  2    4    9     13
  3    9    13     22

Average waiting time = 32.00
Average turn around time = 54.00
```