**BANKERS ALGORITHM**

1. read no of processes(p0,p1,p2,p3)
2. read no of resources (A,B,C,D)
3. read alloc (for p processes and r resources)
4. read max (for p processes and r resources)
5. read avail (for p processes and r resources)
6. calculate need (for p processes and r resources) [NEED = MAX - ALLOC]
7. set all completed processes as 0
8. do while count !=p & process!=-1
9. process no: 'i' will be assigned to 'process' variable .So initialized as -1
10. for each process p , if completion of p is 0 ,set process to i
11. for each resource r , if need > avail (process not selected),set process -1
12. break loop if selected (process!=-1)
13. if selected
14. print 'process +1 selected' (if p=3 ; i=0,1,2,3 ; if i = 2 selected ,3rd process is selected)
15. process stored to safe sequence matrix
16. for each resource r , new alloc = alloc +avail (not i but selected i which is 'process') ,alloc & max = 0 ,completion of 'process' =1
17. if all process selected ,ie count ==p
18. print it is in safe state
19. print safe sequence [i]
20. else print unsafe

**SJF SCHEDULING**

1. read no: of processes
2. read burst time (for each process n )
3. sort according to burst time
4. for each n process (i)
5. for each next process (j=i+1)
6. if bt[i]>bt[j] , swap bt & p
7. define swap function
8. define wt\_tat
9. for first process tat = bt
10. from next process , new tat = tat +new bt && total tat will be updated
11. for first process wt=0
12. from second process , new wt= wt+bt && total wt will be updated

**FCFS SCHEDULING**

1. read no: of processes
2. read burst time (for each process n )
3. sort according to burst time
4. for each n process (i)
5. define wt\_tat
6. for first process tat = bt
7. from next process , new tat = tat +new bt && total tat will be updated
8. for first process wt=0
9. from second process , new wt= wt+bt && total wt will be updated