VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT on

OPERATING SYSTEMS

Submitted by

PARTH RAWAT

(1BM22CS190)

in partial fulfillment for the award of the degree of BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)
BENGALURU-560019
Feb-2025 to June-2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019 (Affiliated To Visvesvaraya Technological University, Belgaum) Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled "OPERATING SYSTEMS – 23CS4PCOPS" carried out by Parth Rawat (1BM22CS190), who is Bonafide student of B. M. S. College of Engineering. It is in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year Feb 2025- June 2025. The Lab report has been approved as it satisfies the academic requirements in respect of a OPERATING SYSTEMS - (23CS4PCOPS) work prescribed for the said degree.

Dr Seema Patil Assistant Professor Department of CSE BMSCE, Bengaluru Dr. Kavitha Sooda Professor and Head Department of CSE BMSCE, Bengaluru

Index Sheet

Sl. No.	Experiment Title	Page No.
1.	Write a C program to simulate the following non-pre-emptive CPU scheduling algorithm to find turnaround time and waiting time. →FCFS → SJF (pre-emptive & Non-preemptive)	1-14
2.	Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time. → Priority (pre-emptive & Non-pre-emptive) →Round Robin (Experiment with different quantum sizes for RR algorithm)	14-28
3.	Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.	28-34
4.	Write a C program to simulate Real-Time CPU Scheduling algorithms: a) Rate- Monotonic b) Earliest-deadline First c) Proportional scheduling	34-40
5.	Write a C program to simulate producer-consumer problem using semaphores	41-45
6.	Write a C program to simulate the concept of Dining Philosophers problem.	45-50
7.	Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.	50-55
8.	Write a C program to simulate deadlock detection	56-59
9.	Write a C program to simulate the following contiguous memory allocation techniques a) Worst-fit b) Best-fit c) First-fit	60-67

10.	Write a C program to simulate page replacement algorithms a) FIFO	68-73
	b) LRU	
	c) Optimal	

Course Outcomes

C01	Apply the different concepts and functionalities of Operating System
C02	Analyse various Operating system strategies and techniques
C03	Demonstrate the different functionalities of Operating System.
C04	Conduct practical experiments to implement the functionalities of Operating system.

Program -1

Question:

Write a C program to simulate the following non-pre-emptive CPU scheduling algorithm to find turnaround time and waiting time.

```
→FCFS

→ SJF (pre-emptive & Non-preemptive)
```

Code:

```
#include <stdio.h>
void main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int arrival[n], burst[n], waiting[n], turnaround[n],
completion[n], response[n];
   printf("Enter Arrival Time and Burst Time for each
process:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d %d", &arrival[i], &burst[i]);
    int currentTime = 0;
    float totalWaiting = 0, totalTurnaround = 0;
printf("\nProcess\tArrival\tBurst\tWaiting\tTurnaround\tResponse\n"
);
    for (int i = 0; i < n; i++) {
         if (currentTime < arrival[i])</pre>
            currentTime = arrival[i];
        completion[i] = currentTime + burst[i];
        turnaround[i] = completion[i] - arrival[i];
        waiting[i] = turnaround[i] - burst[i];
        response[i] = completion[i] - arrival[i];
        totalWaiting += waiting[i];
        totalTurnaround += turnaround[i];
        printf("%d\t%d\t%d\t\d\t\d\n", i + 1, arrival[i],
burst[i], waiting[i], turnaround[i], response[i]);
        currentTime = completion[i];
    printf("\nAverage Waiting Time: %.2f", totalWaiting / n);
```

```
printf("\nAverage Turnaround Time: %.2f\n", totalTurnaround / n); }
```

Result:

```
V TERMINAL

PS C:\Users\Admin\Documents\temp> cd "c:\Users\Admin\Documents\temp\" ; if ($?) { gcc fcfs.c -o fcfs } ; if ($?) { .\fcfs }

Enter number of processes: 4

Enter Arrival Time and Burst Time for each process:

Process 1: 0

7

Process 2: 0

3

Process 3: 0

4

Process 4: 0

6

Process Arrival Burst Waiting Turnaround Response

1 0 7 0 7 7

2 0 3 7 10 10

3 0 4 10 14 14

4 0 6 14 20 20

Average Waiting Time: 7.75

Average Waiting Time: 7.75

Average Turnaround Time: 12.75
```

```
storite a C paragonam to implement good come sient some
algorithm CFCFS).
tine lude (50010, h)
void main()
   int n;
  printf ("Enter number of processes: ");
   Scorf (" 1.d", Orn);
   int arrivaling, bursters, waiting ins, turnaround ins, completion ins;
   printf ("Enter arrival-time and burst time for each proces : In");
  for Cirt i = 0; isn: i++) &
         printf("Process 7.d:", i+ 1);
         scanf ("T.d. 1.d", ForrivolEid; Floorst [13); 9
   int corrent Time = 0;
   float total Waiting = 0, total Turreround = 0;
    printf ("In Process to Arrival to Woiting It Tornaround (");
    for(In+ i= 0; 1 < n; 1++) €
          if Correntime < arrivalci)
                CURSENT I'M # BOM GrivalI)];
              completion [i] = current Time + burst [i];
             burnaround [i] = completion (i) - Grriva(Ei);
               Woiting [i] = burnaround [i] -hurs+ [i];
                Majonse [i] = completion [i] - grrival [i]
                Lotal Turnaround + = turnaround I'J'
                total writing += writing Er 3;
         printf ("7.d) + 7.d x+7.d x+7.d + 7.d +67.d \n", i+1,
                arrivalcis, burst [i], woiting [i], tur norcurdei]
                current Time ~ completion [13;
```

printforn Aurage Waiting Time: 7.24", total waiting 13)
printforn mercage Turnoround Time: 7.24", total waiting 13) Output Enter number of perocesses: 4 Inter securil Time and Bured Time for each perocess: brocess 1: 0 Berocos 2: 0 Buscess 3:0 Process 4: 0 Brown AT 20 Avery Wastery Tim: 7,75 Away Swampion Time: 12.75

=>SJF(Non-preemptive):

Code:

```
#include <stdio.h>
void nonPreemptiveSJF(int n, int at[], int bt[], int ct[], int
tat[], int wt[], int rt[])
{
    int completed = 0, time = 0, min bt, shortest, finish time;
    int remaining_bt[n];
    for (int i = 0; i < n; i++)
        remaining bt[i] = bt[i];
    }
    while (completed < n)
    {
        min bt = 9999;
        shortest = -1;
        for (int i = 0; i < n; i++)
        {
            if (at[i] \le time \&\& remaining bt[i] > 0 \&\& bt[i] <
min_bt)
             {
                min bt = bt[i];
                shortest = i;
            }
        if (shortest == -1)
            time++;
            continue;
        time += bt[shortest];
        remaining_bt[shortest] = 0;
```

```
completed++;
       ct[shortest] = time;
       tat[shortest] = ct[shortest] - at[shortest];
       wt[shortest] = tat[shortest] - bt[shortest];
       rt[shortest] = wt[shortest];
   }
}
void displayTable(int n, int at[], int bt[], int ct[], int tat[],
int wt[], int rt[])
{
   printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\tRT\n");
   for (int i = 0; i < n; i++)
       ct[i], tat[i], wt[i], rt[i]);
}
int main()
{
   int n;
   printf("Enter number of processes: ");
   scanf("%d", &n);
   int at[n], bt[n], ct[n], tat[n], wt[n], rt[n];
   printf("Enter Arrival Time and Burst Time for each
process:\n");
   for (int i = 0; i < n; i++)
    {
       printf("Process %d - Arrival Time: ", i + 1);
       scanf("%d", &at[i]);
       printf("Process %d - Burst Time: ", i + 1);
       scanf("%d", &bt[i]);
   }
```

```
nonPreemptiveSJF(n, at, bt, ct, tat, wt, rt);
displayTable(n, at, bt, ct, tat, wt, rt);
return 0;
}
```

```
Enter number of processes: 4
Enter Arrival Time and Burst Time for each process:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 7
Process 2 - Arrival Time: 8
Process 2 - Burst Time: 3
Process 3 - Arrival Time: 3
Process 3 - Burst Time: 4
Process 4 - Arrival Time: 5
Process 4 - Burst Time: 6
Process AT
                BT
                        CT
                                TAT
                                        WT
                                                RT
1
        0
                                        0
                                                0
2
        8
                        14
                                6
        3
                4
                                        4
                        11
                                8
                                                4
       5
4
                6
                        20
                                15
                                        9
                                                9
```

```
A Moute a C perogram to uniflement shortest too Fine CSIF)
   scheduling (Non-foremplus)
    # include < stdio. h>
   void non Precomptive 33 F Cent n, int at C], int btC], int et[],
                 in tates, int wecz, int ress)
      int completed = 0, time = 0, min_ht, shorter, finish_time;
       int runaining - bit End;
        for lint = 0; ich; i++)
           remaining-bt Cid = bt Cid;
          while (completed kn)
             min-bit = 9119;
          Shortus = -1; . . . .
              for ( int i = 0; i'en; i++)
                 if cat [i] = time X & remaining-ht[i]>0
                 ST bECi3 Lmin-by
                   min -bt Cshortest ) 0
                    completed ++;
                 ct denor exert ? + time;
                   tat I shottes el - ct
                    Shortest = i;
               if (shortest = = -1)
                      time ++; continue; }
```

```
tion r= b tisherks1),
     me maining - bt Catertation
     completed +;
     ct [shortest] = time;
       but Isherics? I = ct[sheries1] - at Espandar];
       wt Estertist? = bat Estert ist? - bl Estortist?;
        re Connect ] = wt Eshortest ];
  void display table lint n, intable 3, inthele 3, interes
                    int 6+13, int well, int HED
          print f (" In Process It AT It BT I ECT It ZATIO WITE,
           for (int ; = 0; ich; i+t)
           printly dit id it id it id it id it id it is in
                attis, Heis, ceris, earis, wetis, ettisi),
           3
       . 4
       No of forgoins:
Buther
                         TAI
             51
                  CI
        AT.
BRALLIN
        0
        8
        3
3
```

=> SJF (Preemptive):

```
#include <stdio.h>
void preemptiveSJF(int n, int at[], int bt[], int ct[], int tat[],
int wt[], int rt[])
{
   int remaining_bt[n];
   int completed = 0, time = 0, min_bt, shortest;
   int flag[n];
   for (int i = 0; i < n; i++)
   {</pre>
```

```
remaining bt[i] = bt[i];
        flag[i] = 0;
    }
    while (completed < n)</pre>
    {
        min bt = 9999;
        shortest = -1;
        for (int i = 0; i < n; i++)
             if (at[i] \le time \&\& remaining bt[i] > 0 \&\&
remaining bt[i] < min bt && flag[i] == 0)</pre>
             {
                 min_bt = remaining_bt[i];
                 shortest = i;
             }
        }
        if (shortest == -1)
        {
            time++;
            continue;
        }
        remaining_bt[shortest]--;
        if (remaining bt[shortest] == 0)
            completed++;
            flag[shortest] = 1;
            ct[shortest] = time + 1;
            tat[shortest] = ct[shortest] - at[shortest];
            wt[shortest] = tat[shortest] - bt[shortest];
            rt[shortest] = wt[shortest];
        time++;
```

```
}
}
void displayTable(int n, int at[], int bt[], int ct[], int tat[],
int wt[], int rt[])
{
   printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\tRT\n");
   for (int i = 0; i < n; i++)
       ct[i], tat[i], wt[i], rt[i]);
}
int main()
{
   int n;
   printf("Enter number of processes: ");
   scanf("%d", &n);
   int at[n], bt[n], ct[n], tat[n], wt[n], rt[n];
   printf("Enter Arrival Time and Burst Time for each
process:\n");
   for (int i = 0; i < n; i++)
       printf("Process %d - Arrival Time: ", i + 1);
       scanf("%d", &at[i]);
       printf("Process %d - Burst Time: ", i + 1);
       scanf("%d", &bt[i]);
   preemptiveSJF(n, at, bt, ct, tat, wt, rt);
   displayTable(n, at, bt, ct, tat, wt, rt);
   return 0;
```

}

```
Enter number of processes: 4
Enter Arrival Time and Burst Time for each process:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 8
Process 2 - Arrival Time: 1
Process 2 - Burst Time: 4
Process 3 - Arrival Time: 2
Process 3 - Burst Time: 9
Process 4 - Arrival Time: 3
Process 4 - Burst Time: 5
Process AT
                BT
                        CT
                                TAT
                                        WT
                                                RT
1
        0
                8
                        17
                                17
                                        9
                                                9
2
        1
                4
                        5
                                4
                                        0
                                                0
        2
                9
                                24
                                                15
                        26
                                        15
                                7
        3
                5
                        10
                                        2
                                                2
```

```
Breempton SIF
void precomptive SSFC lot n, lot of E), int bt E3, int ct[], int fers ],
                    int well, (nt rt E3)
     2
       int remaining - ht End.
       int completed = 0, time = 0, min - ht, shortest;
        set flagens; ..
        for cint i = 0; ikn; itt)
            amoining - bt Ci] = bt Li];
             Flagciz=0;
      while (completed <n)
           min_bt = 9999;
            Shortet =-1;
           for cint 1=0; isn; i++)
               IF COLE is L=time & Fromming - httis 70 90
                        remodeling-becije min-be TT fluxi J=0
                      min-bt= remaining-bti3;
              if cshortest ==-1)
                   time ++;
                   continu;
              ramorning-btsshortes+3--;
```

```
if (remaining-ht Estort st ] == 0)

{

tompleted + t',

flag Estort st ] = 1

ct Estort st ] = 1

but Estort st ] = ct Estort st ] = a* Estort st ];

wt Estort st ] = but Estort st ] = ht Estort st ];

rt Estort st ] = wt Estort st ];

rt Estort st ] = wt Estort st ];

time - tt

}

Quite No. & forces : 4

Process AT BY CT TAT wr RT

1 0 8 17 17 9 9

2 1 4 5 4 0 5

3 2 9 26 24 15 15

4 3 5 10 7 2 2
```

Program 2

Question

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.

- → Priority (pre-emptive & Non-pre-emptive)
- → Round Robin (Experiment with different quantum sizes for RR algorithm)

=> Priority Scheduling (Non-preemptive):

```
#include <stdio.h>
//non-preemptive
void priorityScheduling(int n, int at[], int bt[], int pr[], int
ct[], int tat[], int wt[], int rt[]) {
   int completed = 0, time = 0, min_priority, highest_priority;
   int flag[n];
   for (int i = 0; i < n; i++) {
      flag[i] = 0;
   }
   while (completed < n) {
      min_priority = 9999;
      highest priority = -1;</pre>
```

```
for (int i = 0; i < n; i++) {
            if (at[i] <= time && flag[i] == 0 && pr[i] <</pre>
min priority) {
                min priority = pr[i];
                highest priority = i;
            }
        }
        if (highest priority == -1) {
            time++;
            continue;
        time += bt[highest_priority];
        flag[highest priority] = 1;
        ct[highest priority] = time;
        tat[highest priority] = ct[highest_priority] -
at[highest priority];
        wt[highest priority] = tat[highest priority] -
bt[highest priority];
        rt[highest priority] = wt[highest priority];
        completed++;
    }
}
void displayTable(int n, int at[], int bt[], int pr[], int ct[],
int tat[], int wt[], int rt[]) {
    printf("\nProcess\tAT\tBT\tPriority\tCT\tTAT\tWT\tRT\n");
    for (int i = 0; i < n; i++) {
       bt[i], pr[i], ct[i], tat[i], wt[i], rt[i]);
}
int main() {
    int n:
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int at[n], bt[n], pr[n], ct[n], tat[n], wt[n], rt[n];
    printf("Enter Arrival Time, Burst Time, and Priority for each
process:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d - Arrival Time: ", i + 1);
        scanf("%d", &at[i]);
        printf("Process %d - Burst Time: ", i + 1);
        scanf("%d", &bt[i]);
        printf("Process %d - Priority: ", i + 1);
        scanf("%d", &pr[i]);
```

```
}
priorityScheduling(n, at, bt, pr, ct, tat, wt, rt);
displayTable(n, at, bt, pr, ct, tat, wt, rt);
return 0;
}
```

```
Enter number of processes: 4
Enter Arrival Time, Burst Time, and Priority for each process:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 4
Process 1 - Priority: 2
Process 2 - Arrival Time: 0
Process 2 - Burst Time: 10
Process 2 - Priority: 1
Process 3 - Arrival Time: 0
Process 3 - Burst Time: 3
Process 3 - Priority: 3
Process 4 - Arrival Time: 0
Process 4 - Burst Time: 12
Process 4 - Priority: 4
Process AT
                вт
                        Priority
                                       CT
                                                TAT
                                                        WT
                                                                RT
1
       0
                4
                                        14
                                                14
                                                        10
                                                                10
       0
                                                                0
                10
                        1
                                        10
                                                10
                                                        0
       0
                3
                        3
                                        17
                                                17
                                                        14
                                                                14
       0
               12
                        4
                                        29
                                                29
                                                        17
                                                                17
```

```
Or Privary Scheduler aposething
    sindule <std10.h>
    void priority scheduling cont n , int alts, int. hl [7, int pris,
    int ot Es; int totEs, int well, int ve Es) f
    int complebed = 0 , time = 0, min - priority , higher-pring
     int floy End;
        For lint is o; ich ; it+) f
             flag ci3 =0;
         while Completed and &
             min - priority = 9599;
highest - priority = -1;
          for What 1:00; ich; ite) {
               if latery to time & + flegen = 20
              TE posis min-priority of
                  min-polonity = sessy prell;
                  highest priority = etg i;
           if Chiquest - priority ==- 1) f ...
              bime ++;
              continue;
         dime += bt [highest - priority];
          Flag Chighest - privarily ]=1;
         Ct [ highest = priority 3 = c6 [ highest = priority 3 = at Orighest = priority ]
              wt [Highest - priority] too [ highest-priority] - be Daybolamenty]
         rt Dhighest -probpity ] = wt [ highest -priority ]
         completed ++;
```

```
wid display Table circl n int obly, int p+ cj rint cb[],
          int total, int well, intrestil
    printer infracess 16A7 16BT 16 Priority 16 CT
                16 TAT YEWILL RT LA OJ.
    forline : = 0; / (n; i+1) (
        points conditionated to year to year of her
          j+1, a(t), b(c)], c+C;], (a(E)], w(E)], n(E)
      4
 int maint) {
   int n',
   printect smer the number of processis;
   5 can ( C'Yd", 47);
    Int at End, he end, pr end, et Dol jear in a wood their
in mainths
  int n'
  prints 1º Enser Accive Flow, burst time and Priority for
eash process s (m);
  For ( Int 1 = 0 ; isp; 1+1) &
     printFL" frocess yed - arrival Time : ",i+1).
     Scenf [" " d" , Sor Ci ]);
     point F(" Viocess I.d - burst 71 mms: " , 171);
      scorf( " or ht[i]);
      printft" Process . "I'd - eriotity " it 1);
     · scorf ("I'd", &por [i]);
     priority Scheduling (n, is 1, it, pr, ct, tat, we, rt);
     display Tohle (n) at , price, tot, wit , rt); reform out
```

```
Outful.
Exten munder of formers 4
Enter AT, BT, and Briefity from each foreces
Consent At 0 -
      BT. 4.
Pic
     Brionto 2
12 AL O
     BT to
P3 : AT:0
      BT : 3
     AT O
      BT 12
              Print CT TAT WT
      0
                                  .10
 2
      0
          10
                                  0
 3
          3
```

=> Priority Scheduling (Preemptive):

```
return highest;
}
void priorityScheduling(struct Process p[], int n) {
    int currentTime = 0, completed = 0;
    float totalWaitingTime = 0, totalTurnaroundTime = 0;
    for (int i = 0; i < n; i++) {
       p[i].remainingTime = p[i].burstTime;
   while (completed < n) {</pre>
        int idx = findHighestPriority(p, n, currentTime);
        if (idx == -1) {
            currentTime++;
            continue;
       p[idx].remainingTime--;
       currentTime++;
        if (p[idx].remainingTime == 0) {
            completed++;
           p[idx].completionTime = currentTime;
           p[idx].turnaroundTime = p[idx].completionTime -
p[idx].arrivalTime;
           p[idx].waitingTime = p[idx].turnaroundTime -
p[idx].burstTime;
            totalWaitingTime += p[idx].waitingTime;
            totalTurnaroundTime += p[idx].turnaroundTime;
       }
    }
printf("\nProcess\tArrival\tBurst\tPriority\tCompletion\tTurnaround
\tWaiting\n");
    for (int i = 0; i < n; i++) {
       p[i].arrivalTime, p[i].burstTime,
              p[i].priority, p[i].completionTime,
p[i].turnaroundTime, p[i].waitingTime);
   }
   printf("\nAverage Waiting Time: %.2f", totalWaitingTime / n);
   printf("\nAverage Turnaround Time: %.2f\n", totalTurnaroundTime
/ n);
```

```
int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    struct Process p[n];

    printf("Enter Arrival Time, Burst Time, and Priority (lower number = higher priority) for each process:\n");
    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        printf("Process %d: ", p[i].id);
        scanf("%d %d %d", &p[i].arrivalTime, &p[i].burstTime,
        &p[i].priority);
     }
     priorityScheduling(p, n);
    return 0;
}</pre>
```

```
Enter number of processes: 4
Enter Arrival Time, Burst Time, and Priority (lower number = higher priority) for each process:

Process 1: 0
5
2
Process 2: 0
3
1
Process 3: 0
8
3
Process 4: 0
2
4

Process Arrival Burst Priority Completion Turnaround Waiting
1 0 5 2 8 8 8 3
2 0 3 1 3 3 0
3 0 8 3 16 16 16 8
4 0 2 4 18 18 16

Average Waiting Time: 6.75

Average Waiting Time: 6.75

Average Waiting Time: 6.75

Average Waiting Time: 6.75
```

```
continut,
  peldre I . remaining Time - -;
   arrest Time ++;
  p Cide I . completion Time = current Time;
   paidri. burnaround Time =
                                   pridal . completion Time -
                                    p cida], arrivol Time;
   count fristed. Endring in antipopulation. Earling
    · p. Eidze ] . Woiting Time = p Lidze ] . turnoround Time - priox ?
      total waiting Time t = plidry worting Time
      total Turvaround Time += & Eid x I . thernanound Time;
   3
  printf (" ) n frocess > 6 Arrival 16 Burst 16 Priority > 6 Completion
           16 Turner and 14 Weiting in "),
   for sint i = 0; i'ch; i++) &
       peint f("1.d 14 1.d 1t 1.d 1t 1.d 1t 1.d 1t 1.d 1n", peilid,
                p Ci3. oreival Time, p Si 3. burst Time, pci3 provity,
                gril-competintime, pris turnoround time, pril.
                                   woiting Time;
   prints ("In " Average waiting time: ". 27", well Torrerowth
  privilet "In Average Turneround Time : 1.2 10");
 3
int main () {
    int ni
    printf (" Enter rumber of processes: ");
    good ("/d", an);
   struct Process pend;
      printf ("Enter Arrival Time, burst time and Primits
```

```
Claver number = higher priority) for each process:
       for Cin: = 0; / = n; / ++) &
          pristid = in li
         print tillous 2.d : 11 , psis .d);
        sign fillidy, d' d 1 x p 5 i 3 . arrival limi, tipe.
            April . pr Drity);
         priority Schodulies (p, n);
         (duno;
                                      Townson way
author .
                                                  UT
                            Completen
                   PHIPM
 Brown AT
                                        8
                                                   3
                              80
              5
        D
                                                   0
                              3
              3
        0
                                       16
                                                  8
                             16
                                                  16
                                       18
                             18
```

=> Round Robin:

```
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int wt[],
int quantum) {
    int rem_bt[n];
    for (int i = 0; i < n; i++) {
        rem_bt[i] = bt[i];
        wt[i] = 0;
        wt++;
    }
    int t = 0;
    while (1) {
        int done = 1;
        for (int i = 0; i < n; i++) {</pre>
```

```
if (rem bt[i] > 0) {
                done = 0;
                if (rem bt[i] > quantum) {
                     rem bt[i] -= quantum;
                     //++quantum;
                     t += quantum;
                 } else {
                     t += rem bt[i];
                     wt[i] = t - bt[i];
                     rem bt[i] = 0;
                 }
            }
        if (done) break;
    }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[],
int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}
void findAvgTime(int processes[], int n, int bt[], int quantum) {
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt, quantum);
    findTurnAroundTime(processes, n, bt, wt, tat);
    int total wt = 0, total tat = 0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround
Time\n");
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
```

```
total tat += tat[i];
        printf("%d\t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i],
tat[i]);
    }
    printf("\nAverage Waiting Time: %.2f", (float)total wt / n);
    printf("\nAverage Turnaround Time: %.2f\n", (float)total tat /
n);
}
int main() {
    int n, quantum;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int processes[n];
    int burst time[n];
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &burst time[i]);
    }
    printf("Enter time quantum: ");
    scanf("%d", &quantum);
    findAvgTime(processes, n, burst time, quantum);
    return 0;
}
```

```
Enter number of processes: 4
Enter burst time for process 1: 10
Enter burst time for process 2: 5
Enter burst time for process 3: 7
Enter burst time for process 4: 3
Enter time quantum: 4
                       Waiting Time Turnaround Time
Process Burst Time
1
       10
2
       5
                       15
                                      20
                       16
                                      23
4
       3
                       12
                                      15
Average Waiting Time: 14.50
Average Turnaround Time: 20.75
```

```
Round Robin.
Hirolude 45thio. h)
void find uniting time (Int process 63, int no int he63,
            int wich link quertum) e
          int remembers;
         for (Int 1 = 0; 16h; 1++) f
          rem _bt fil + htfij;
            wtCileo;
      3
      int 6=0;
       while (1) (
          int done = 1;
            For ( int 1 = 0; 1:4n; 11+) {
                1 + ( vem- bl E/2 > 0) {
                  done = 0; ....
                 if (rem-htcil Iquantum) f
                        rem_blicio = quent um;
                       t 7 = quantum;
                        3 160 -2
                       6+= r+m_bt617;
                       7 0/56 6
                        + +> rim -becil;
                          wtcil = E- striz;
                           Fem- bt [i]-0:
                     3 3
                 it chone ) break;
                33
 void find Ture Mound Time Cint processesses, int n,
                           int htc), int wt [], int to 163)
    FOR ( but i= 0; i < n; i++) & tacis = Heily w+6:7,33
```

void find any Time Cint processis [] , Int no inthosy int quentum) & int woons, totens; find westing Time (processe the holimbe greens Find Torn Around Time Grocusis, n. bb, wt int total - wt = 0, total - tot= 0 printf ("In Process It Burst Time It working Time Time In ")' for Cinti= 0; ich; it+) & total- wt += wt[i]; botcl-late text [1]; printf (1/1. detid tet 1. d to the 1. d in if rough lotsis, we til, fet til); printfent banange walting Time: 1.2 1" 16/mg point f ("In Average Turnary and Time : Y .. 2 F. (flact) total -tat /n); Outlest Serve quanton 4 WT TAT BI 20 23 Average WT 14.50 Annag 7A7 20.75

Program 3

Question

Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.

=> Multilevel queue Scheduling

```
#include <stdio.h>
#define TIME QUANTUM 2
typedef struct {
    int pid, burst time, arrival time, queue;
    int waiting time, turnaround time, response time,
remaining time;
} Process;
void sort by arrival(Process p[], int n) {
    Process temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (p[i].arrival time > p[j].arrival time) {
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
void round robin(Process p[], int n, int *time) {
    int done, i;
    do {
        done = 1;
        for (i = 0; i < n; i++) {
            if (p[i].remaining time > 0) {
                done = 0;
                if (p[i].remaining time > TIME QUANTUM) {
                     *time += TIME QUANTUM;
                     p[i].remaining_time -= TIME_QUANTUM;
                 } else {
                     *time += p[i].remaining time;
                     p[i].waiting time = *time - p[i].arrival time -
p[i].burst time;
                     p[i].turnaround_time = p[i].waiting_time +
p[i].burst time;
                     p[i].response time = p[i].waiting time;
                     p[i].remaining time = 0;
                 }
            }
    } while (!done);
void fcfs(Process p[], int n, int *time) {
    for (int i = 0; i < n; i++) {
        if (*time < p[i].arrival time)</pre>
```

```
*time = p[i].arrival time;
        p[i].waiting time = *time - p[i].arrival time;
        p[i].turnaround time = p[i].waiting time + p[i].burst time;
        p[i].response time = p[i].waiting time;
        *time += p[i].burst time;
    }
}
int main() {
    int n, i, time = 0;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    Process p[n], system processes[n], user processes[n];
    int sys_count = 0, user count = 0;
    for (i = 0; i < n; i++) {
        printf("Enter Burst Time, Arrival Time and Queue of P%d: ",
i + 1);
        p[i].pid = i + 1;
        scanf("%d %d %d", &p[i].burst time, &p[i].arrival time,
&p[i].queue);
        p[i].remaining time = p[i].burst time;
        if (p[i].queue == 0)
            system processes[sys count++] = p[i];
        else
            user processes[user count++] = p[i];
    sort by arrival(system processes, sys count);
    sort by arrival (user processes, user count);
    printf("\nQueue 1 is System Process\nQueue 2 is User
Process\n");
    round robin(system processes, sys count, &time);
    fcfs(user processes, user count, &time);
    Process final list[n];
    int index = 0;
    for (i = 0; i < sys count; i++)
        final list[index++] = system processes[i];
    for (i = 0; i < user count; i++)
        final list[index++] = user processes[i];
    printf("\nProcess\tWaiting Time\tTurn Around Time\tResponse
Time\n");
    float avg wt = 0, avg tat = 0, avg rt = 0;
    for (i = 0; i < n; i++) {
```

```
printf("%d\t%d\t\t\t%d\n", final list[i].pid,
final list[i].waiting time, final list[i].turnaround time,
final list[i].response time);
        avg wt += final list[i].waiting time;
        avg tat += final list[i].turnaround time;
        avg rt += final list[i].response time;
    avg wt /= n;
    avg_tat /= n;
    avg rt /= n;
    float throughput = (float)n / time;
    printf("\nAverage Waiting Time: %.2f", avg wt);
   printf("\nAverage Turn Around Time: %.2f", avg tat);
    printf("\nAverage Response Time: %.2f", avg rt);
    printf("\nThroughput: %.2f", throughput);
    return 0;
}
```

```
Enter number of processes: 4
Enter Burst Time, Arrival Time and Queue of P1: 2
1
Enter Burst Time, Arrival Time and Queue of P2: 1
Enter Burst Time, Arrival Time and Queue of P3: 5
Enter Burst Time, Arrival Time and Queue of P4: 3
2
Queue 1 is System Process
Queue 2 is User Process
Process Waiting Time
                        Turn Around Time
                                                Response Time
                        2
1
        0
                                                0
2
        2
                        3
                                                2
3
                                                3
        3
                        8
4
        8
                                                8
                        11
Average Waiting Time: 3.25
Average Turn Around Time: 6.00
Average Response Time: 3.25
Throughput: 0.36
```

```
a Meltiduel queue exhauling
```

```
# include KSHU, o. h)
4 define TIME_QUANTUM 2
by rdef stover &
     int pid , hurst - time , actival - time , queue;
     int waiting-time, turnaround-time, response-time,
          remaining - ther;
  3 froms
 void some - by -arrivel (frocess pl), int n) {
    110cess trop;
     for (int i =0; ith); i++) {
     for (j= i+1; j4h; j++) 2
          if (priliarriveltime > prijlarriveltime) (
              termy = pril;
              pcij = pcjs;
              PEJJ - 61mp; 1774
  void round-kabin ( Brokess pcz, int n , int time) {
          int done ii;
          do 4
           dor== 1;
            forli=0 , i4n; i++) {
               ir Cpcilifemaining Time >0 - townsy Tember Gym
               done : D !
              ir ( phil, remaining = time 274ME avariants 1 $
                Himet - BRASH TO CAMON
                 "time+= TIME - GUANDOM;
                 pci) . waiting - time = time - pci Tarrialism;
                 pri Teturnaround time
                 pcil. remaining - bime = 0;
              33
```

```
7 while (!done);
void five Clours pET, int nint + for &
     for lint is 0, ish; int) f.
       If C'hone < ptilarrival time)
            times = pcis arrival = Lipnes ....
        pers. waiting - time = time - persencial-line.
        1613 terroground time = pt. 1, waiting ations + pt.
        pciling-time = cciz. waiting-time;
         time to psyl burst bing
in main() (
   int with time = 0; --
   product Enter member of processes: ")
   510xf Cey, d", 4.10);
   Proces pEPT, System - processes EMT, use - process EBJ;
   100 sq-comt = 0 , user - count = 0;
   14 (1=0; 14h; 17+2 f.
     print fill Enter purst Time, Arrival Time and
          BULL OF PY.d:" , 1+13 .
          prid. remaining - time = px 13, horse time:
      $3 return 0:
      Y
```

Jether -	Enter mumbe	n of torocons	14:4		
	Enter B7, A			1:20	,
	Enter MT,	at anol on	ere of P	2:10	2
	unter BT,	AT and qu	as of T	3: 5 0	-
	Enter BT,	AT and only	eu og P	5: 3 0	-2
	u 1 b ageton 1				
Own	2 ; was P	noces			
Bures	albitis Sinc	Someon	Sire	Rochonce	ci.
1	. 0 .	. 2		0	WH-C
2	2	3 .		2	
3	3	8		3	
4	8		21 11		
		10		1 1	
Averner.	WT * . 3.25	***		Tayonay	
	0 TAT 6.00			10.18	
	R7: 3.2		400		
	gaspie 0.30		12.0		
	71-64-0				

Question

Write a C program to simulate Real-Time CPU Scheduling algorithms:

- -> Rate- Monotonic
- -> Earliest-deadline First
- -> Proportional scheduling

=> Rate Monotonic Scheduling

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
   int id;
   int period;
   int execution_time;
   int next_deadline;
   int executed;
} Task;

int compare_tasks(const_void *a, const_void *b) {
```

```
return ((Task *)a) ->period - ((Task *)b) ->period;
}
void rate monotonic scheduling (Task tasks[], int num tasks, int
total time) {
    qsort(tasks, num tasks, sizeof(Task), compare_tasks);
    /*
    for(int i = 0; i < num_tasks; i++)</pre>
        printf("Task %d: %d %d\n", tasks[i].id,
tasks[i].execution time, tasks[i].period);
    for (int i = 0; i < num tasks; i++)
        tasks[i].next deadline = tasks[i].period;
    printf("Time\t");
    for (int i = 0; i < num tasks; i++)
        printf("Task %d\t", tasks[i].id);
    printf("\n");
    for (int current time = 0; current time < total time;
current time++)
        printf("%d\t", current_time);
        int executed task = -1;
        for (int i = 0; i < num tasks; i++)
            if (current time % tasks[i].period == 0)
                tasks[i].next deadline = current time +
tasks[i].period;
                tasks[i].executed = 0;
            }
            if (current time < tasks[i].next deadline)</pre>
                if(tasks[i].executed < tasks[i].execution time)</pre>
                     executed task = i;
                     tasks[i].executed++;
                     break;
                 }
            }
        if (executed task !=-1)
            for (int i = 0; i < num tasks; i++)
```

```
{
                 if (i == executed task) {
                     printf("Exec\t");
                 } else {
                     printf("\t");
             }
        } else {
              for (int i = 0; i < num_tasks; i++) {</pre>
                 printf("\t");
        printf("\n");
    }
}
int main() {
    Task tasks[] = {
        {1, 20, 3},
        \{2, 5, 2\},\
        {3, 10, 2}
    };
    int num_tasks = sizeof(tasks) / sizeof(tasks[0]);
    int total time = 20;
    rate_monotonic_scheduling(tasks, num_tasks, total_time);
    return 0;
}
```

```
Task 2 Task 3 Task 1
          Exec
1
2
3
4
5
6
7
8
9
          Exec
                   Exec
                   Exec
                            Exec
          Exec
          Exec
                            Exec
                            Exec
          Exec
11
          Exec
12
13
 14
 15
          Exec
16
          Exec
17
 18
19
```

```
Rate monotoni schoolderig
# Indust 451800.n)
      Int id, executiontime, priod; ?
grovet took &
     void gont Task sc struct Task baskers, in 1 hos
for (int in 0; i-n-1-1; it+)

For (i - 0; i-n-1-1; i++)
                i for tacks Is 7 por load > tas his Ego 17 pring
           Struct Task timp = tasks=j3;
                 basASEj3 " bosks Ej+13;
                 baks Ej+17 = temp;
 Int main () {
      int h, +:me = 20;
       print + (" Eater eummher of lasks: ")
      sconf (" 1 d", & h);
      for (int i=0; i ch; iar) {
          printf("Tage 1.1. execution time and
            period: ", 171);
          Scon F ("d' I'd", STOCKES [ ] Jerrerution Time,
                             Trasks [1] - period);
                      tesastil.10-1+1;
          gort Tocke ( +GGRs , n);
          print f ("RMS In");
             for first t = 0; t < time; t+24
                  for (int := 0; 12n; 1+ ) {
                    if Ct 7 tasks C1 Juniod Ktaskers
```

```
& printer time yd : Tack ydn", t, tosks [i] id);
                 exercised =1;
                  brok;
                 printf b"time yet - Idle in", +);
                ritury 0',
Delped
                 Jack 2
                                     Jask 1
         0
                  Inc
                  Exec
         3
                            Sicoc
        4
                 Exer
        5
                                    nou
        9
       10
               there
       13
       14
                tour
       15
       16
      17
      18
```

=> Earliest Deadline First

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
   int id;
   int period;
   int execution_time;
   int deadline;
```

```
int executed;
} Task;
int compare tasks(const void *a, const void *b) {
    return ((Task *)a)->deadline - ((Task *)b)->deadline;
void earliest deadline first scheduling(Task tasks[], int
num_tasks, int total_time) {
    printf("Time\t");
    for (int i = 0; i < num tasks; i++)
        printf("Task %d\t", tasks[i].id);
    printf("\n");
    for (int current_time = 0; current time < total time;</pre>
current time++) {
        printf("%d\t", current time);
        int executed task = -1;
        for (int i = 0; i < num_tasks; i++) {</pre>
             if (current time % tasks[i].period == 0) {
                 tasks[i].deadline = current time + tasks[i].period;
                 tasks[i].executed = 0;
             }
        qsort(tasks, num tasks, sizeof(Task), compare tasks);
        for (int i = 0; i < num_tasks; i++) {</pre>
             if (current time < tasks[i].deadline &&</pre>
tasks[i].executed < tasks[i].execution time) {</pre>
                 executed task = i;
                 tasks[i].executed++;
                 break;
             }
        if (executed task !=-1) {
             for (int i = 0; i < num_tasks; i++) {</pre>
                 if (i == executed task) {
                     printf("Exec\t");
                 } else {
                     printf("\t");
             }
        } else {
             for (int i = 0; i < num tasks; <math>i++) {
                 printf("\t");
             }
```

```
Time
        Task 1 Task 2 Task 3
0
        Exec
1
        Exec
2
                Exec
3
4
                Exec
                        Exec
                Exec
6
        Exec
                         Exec
8
                         Exec
9
10
        Exec
11
        Exec
12
                        Exec
13
                Exec
14
15
                         Exec
16
                Exec
17
18
19
```

Question

Write a C program to simulate producer-consumer problem using semaphores

=> Producer Consumer

```
#include <stdio.h>
int x = 1, mutex = 1, full = 0, empty = 3;
void wait(int *S)
    (*S)--;
}
void signal(int *S)
    (*S)++;
void producer()
    wait(&mutex);
    if (empty > 0)
        wait(&empty);
        signal(&full);
        printf("Item produced: %d\n", x++);
    } else {
        printf("Buffer is Full\n");
    signal(&mutex);
void consumer() {
    wait(&mutex);
    if (full > 0) {
        wait(&full);
        signal(&empty);
        printf("Item Consumed: %d\n", --x);
        printf("Buffer is Empty\n");
    signal(&mutex);
}
int main() {
```

```
int ch;
printf("1. Produce\n2. Consume\n3. Exit\n");
while (1) {
    printf("Enter Choice: ");
    scanf("%d", &ch);
    switch (ch) {
        case 1: producer(); break;
        case 2: consumer(); break;
        default: return 0;
}

Output:
```

```
1. Produce
2. Consume
3. Exit
Enter Choice: 2
Buffer is Empty
Enter Choice: 1
Item produced: 1
Enter Choice: 1
Item produced: 2
Enter Choice: 1
Item produced: 3
Enter Choice: 1
Buffer is Full
Enter Choice: 1
Buffer is Full
Enter Choice: 2
Item Consumed: 3
Enter Choice: 2
Item Consumed: 2
Enter Choice: 2
Item Consumed: 1
Enter Choice: 2
Buffer is Empty
Enter Choice:
Buffer is Empty
Enter Choice: 2
Buffer is Empty
Enter Choice: 3
```

```
Sining Philappies of Broducer arrange
Broduces Consumes
Hirelade Zadio In?
int Y=1, motex=1, full = 0, compty = 3;
void wollcint's)
    C+5>-:
void signal (int * S)
     ( s) ++; y
 void product() &
      wait ( of mutus);
         is (compty 30)
          wit (Sempty)?
Pignal (Stoll);
            bring the brogging : Ad la 1 x ++ );
             printf("Bother is full m");
            signal (of moder);
       void consumer D. (
            wast (4 moutex);
             4 (FULL 20) (
                 weit ( 4 fully;
                Signal ( Stimpty);
```

```
phod printfl'Item consumed: 1.d. 14", -- 10);
         prontforby for is musty in " );
      signal ( insuled);
   int mainth
       Int ch:
       print ["1. Produce In 2. Consumor In 3. Exit In")
         printel "Enter charce");
        Scarf 17.d", 9(h);
        Switch (ch) 5
          cose 1: producer (); break;
         case 2: (onsumers) ; boulk',
          default return 0;
Outpest 1. Brooker
    2 Amount
    3 900
Enter Choice 1
 The was produced ! !
Enter those !
there produced 2
Enter choice 1
 Hem produced 3
 Enter Chaire' 1
  Butter is fell
 Enter choice: 2
Hern congeneed: 2
```

Question

Write a C program to simulate the concept of Dining Philosophers problem.

=> Dining Philosophers

Code

//PTHRED AND SEMAPHORE LIBRARY ONLY WORK IN CODEBLOCKS, NOT VSC #include <pthread.h>

```
#include <semaphore.h>
#include <stdio.h>
#include <unistd.h>
#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N
int state[N];
int phil[N] = \{0, 1, 2, 3, 4\};
sem t mutex;
sem t S[N];
void test(int phnum) {
       if (state[phnum] == HUNGRY && state[LEFT] != EATING &&
state[RIGHT] != EATING) {
        state[phnum] = EATING;
        sleep(2);
        printf("Philosopher %d takes fork %d and %d\n", phnum + 1,
LEFT + 1, phnum + 1);
        printf("Philosopher %d is Eating\n", phnum + 1);
        sem post(&S[phnum]);
    }
}
void take fork(int phnum) {
    sem wait(&mutex);
    state[phnum] = HUNGRY;
    printf("Philosopher %d is Hungry\n", phnum + 1);
    test (phnum);
    sem post(&mutex);
    sem_wait(&S[phnum]);
    sleep(1);
}
void put fork(int phnum) {
    sem wait(&mutex);
    state[phnum] = THINKING;
    printf("Philosopher %d putting fork %d and %d down\n", phnum +
1, LEFT + 1, phnum + 1);
    printf("Philosopher %d is thinking\n", phnum + 1);
```

```
test(LEFT);
    test(RIGHT);
   sem post(&mutex);
}
void* philosopher(void* num) {
    while (1) {
        int* i = (int*)num;
        sleep(1);
        take fork(*i);
        sleep(0);
        put_fork(*i);
   }
}
int main() {
   int i;
   pthread_t thread_id[N];
    sem init(&mutex, 0, 1);
    for (i = 0; i < N; i++) {
       sem init(\&S[i], 0, 0);
    for (i = 0; i < N; i++) {
                pthread_create(&thread_id[i], NULL, philosopher,
(void*)&phil[i]);
       printf("Philosopher %d is thinking\n", i + 1);
    for (i = 0; i < N; i++) {
       pthread join(thread id[i], NULL);
   return 0;
Output:
```

```
C:\Users\Admin\Documents\t X
Philosopher 4 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 1 is Hungry
Philosopher 3 is Hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 5 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 2 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 4 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 1 is Hungry
Philosopher 5 putting fork 4 and 5 down
Philosopher 5 is thinking
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 3 is Hungry
Philosopher 5 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
```

```
Dancy Philosopher
    Hincluck Epthroad in >
    # include compensare.ht
    Minelude 4 stdio. b.
    windlide < unishliky
    udefine LEFT (phonom + 4) 1 N
    Hother RIGHT (phinontl) "N
   Holfing NG
   # dix or TUNKING
  # differ HUNDRY !
  # diffine BATIMA O.
  jet stoll INZ;
  int phil[N] = Co, 1,2,3,43
 sime a mobile;
  Som t SCAS;
 void but Cont phonoun) &
      if (ital present == HUNDAY $ $ glate CLEFT ]
    TT STATE CRIGHT? 1 = GATING ) &
             State [ph mone] = Earling;
              Ship (2);
              print F( "Philosopher "d & Edbing In john or +1)
              Sim-post (955 Ephono I);
word take _fork Cint phrum) {
         sim - wort ( & muters);
         Litel . [phown] = HUNGRY;
         print P M Philosoph er idias Horge y In's prhown t
```

```
tost (phonum):
   som - post (& mouter);
   com - west C X S I phowm 1);
   5/00 (1); 3
  void put - fork (int phrom) f
         fine with cameting);
             State Copprom3 - THINKING,
 void philosophic printe (" enverger "de prising for a 7.8
  and id down in prince 1, LF FT +1, procin +1);
      princet ("Prilosop her 7 d is trinking in", phonon + 1);
       test LLEPO:
      bist (A1647);
     Sim-post (Smuter)
 void * philosopher (void * rum) {
    while (1) 9
        1309 $ 1 = (1200 * ) He sons
           steepell;
           take fork (1);
           sleep 10);
Output Philosopher 4 is hungry
      Britisafter Sed fully book 4 and 5 down
     Orderlytes 5 in thinking
     thelogher to take book 3 and 4
     Belonger 4 is sature
     Bhillosphian I es lowgray
```

Question

Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

=> Banker's Algorithm / Deadlock Avoidance

```
#include <stdio.h>
#include <stdlib.h>
int condition(int **need, int *work, int i, int m)
{
    for (int j = 0; j < m; j++)</pre>
```

```
if (need[i][j] > work[j])
            return 0;
    return 1;
int safety(int m, int n, int **allocated, int **max, int
*available, int *sequence)
    // Need Matrix
    int **need = (int**) malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++)
        need[i] = (int*) malloc(m * sizeof(int));
        for (int j = 0; j < m; j++)
            need[i][j] = max[i][j] - allocated[i][j];
    // Work array
    int *work = (int*) malloc(m * sizeof(int));
    for (int i = 0; i < m; i++)
        work[i] = available[i];
    // Finish array
    int *finish = (int*) malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
        finish[i] = 0;
    int safeIndex = 0;
    int changed;
    do {
        changed = 0;
        for (int i = 0; i < n; i++)
            if (!finish[i] && condition(need, work, i, m))
            {
                for (int j = 0; j < m; j++)
                    work[j] += allocated[i][j];
                finish[i] = 1;
                sequence[safeIndex++] = i;
```

```
changed = 1;
            }
    } while (changed);
    for (int i = 0; i < n; i++)
        if (!finish[i])
            return 0;
    return 1;
int main()
    int n, m;
   printf("Enter number of processes and resources (n x m order):
   scanf("%d",&n);
    scanf("%d", &m);
    // Allocation Matrix
    printf("Enter Allocation Matrix:\n");
    int **allocated = (int **) malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++)
        allocated[i] = (int*) malloc(m * sizeof(int));
        for (int j = 0; j < m; j++)
            scanf("%d", &allocated[i][j]);
    }
    // Max Matrix
    printf("Enter Max Matrix:\n");
    int **max = (int **) malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++)
    {
        max[i] = (int*) malloc(m * sizeof(int));
        for (int j = 0; j < m; j++)
            scanf("%d", &max[i][j]);
    // Available Matrix
    printf("Enter Available matrix:\n");
```

```
int *available = (int *) malloc(m * sizeof(int));
for (int i = 0; i < m; i++)
    scanf("%d", &available[i]);
// Sequence Matrix
int *sequence = (int *) malloc(n * sizeof(int));
int safe = safety(m, n, allocated, max, available, sequence);
if (safe)
    printf("System is in a Safe State.\nSafe Sequence: ");
    for (int i = 0; i < n; i++)
        printf("P%d\t", sequence[i]);
    printf("\n");
}
else
{
    printf("System is not in a Safe State.\n");
return 0;
```

```
Enter number of processes and resources (n x m order): 5 3
Enter Allocation Matrix:
0 1 0
200
3 0 2
2 1 1
002
Enter Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Available matrix:
3 3 2
System is in a Safe State.
Safe Sequence: P1
                                        P0
                                               P2
                               P4
```

```
on Brimber is algoriche
      Hardlade Ladions
      # include accellining
       int condition Cont * " need, int " work, int , lot on)
            for (in+ j = 0, j = m ; j ++)
                if eneed I SEj] > work Ej 3)
                  Atom O; 3
                return 1; 3
     in + Sufity eint my into , int + + alexaled job + + maximile
                                                     int syund
            in + to need = (in+++) needle (h+ size of (ion +))
             for linti = 0, 14n; 171)
                med ci3 = linte + ) maloc (m + size of (h+1);
                  For (int 1=0, 140) H+)
                    herd Liz = Got ") malloc (me size of (ind);
                     For con j=0; jem; jea)
                       head Listis = morsty 3 -allocated sipility
        int sais Index = 0;
        int changed;
            charged = 0
            for Mot i " O, ich ; jar )
```

```
if Clfinish cis & an condition (need, work, 1, m))
 for (int j=0; j 4 mo; j++)
         workesit= allocated Eilesi; }
        Finish Ei3 =1 :
       Signince Esofe Irole ++ ] = 1:
        Changed = 1, 9 7
  I while (changed);
   Portion 1=0; i'an; in+)
    if ( Anish Eis) f
      Ideno; & I returnly 7
int main()
2
   int non;
   printf ("Enter number of processes and resources (ham)
                                                order):"
   Scort (" V.d", Vn);
   5008(1/d", 4m);
    printf ("Enter Allocation Matrix: (n");
    int + alterited = (nt + e) miglior (n' give of cint ));
     for Cirtiso ; ich; He) &
       allowed Lis = Got + ) malloc com + sixtof (in));
          for (in+ j=0; j < no; j++){
            sconf cord", Fallocuted cil cj3); 33
      pertf ("Enter May Maret rin : In");
       int + max = (in+ 12) malloc (n+ size of (in+ >))
        mone Eiz = (int*) malloc (a " sixofict"))-
          For Cint i= Dj i Ln; in-
         1 more is = (int) molloc (m + size of cinty);
```

Under Anadolic motion

3 3 2

System is in Sote State
Sofe signines: 11 13 14 10 P2

Question

Write a C program to simulate deadlock detection

=> Deadlock Detection

```
#include <stdio.h>
#include <stdbool.h>
#define P 5
#define R 3
int main() {
    int finish[P] = \{0\};
    int work[R];
    int need[P][R] = {
        \{7, 5, 3\},\
        {3, 2, 2},
        {9, 0, 2},
        {2, 2, 2},
        {4, 3, 3}
    };
    int allocation[P][R] = {
        {0, 1, 0},
        {2, 0, 0},
        {3, 0, 2},
        {2, 1, 1},
        {0, 0, 2}
    };
    int available[R] = \{3, 3, 2\};
    for (int i = 0; i < R; i++) {
        work[i] = available[i];
    }
    bool deadlock = false;
    int count = 0;
    while (count < P) {
        bool found = false;
        for (int p = 0; p < P; p++) {
             if (finish[p] == 0) {
                 bool canFinish = true;
                 for (int r = 0; r < R; r++) {
                     if (need[p][r] - allocation[p][r] > work[r]) {
                          canFinish = false;
                         break;
                     }
                 }
```

```
if (canFinish) {
                    for (int r = 0; r < R; r++) {
                        work[r] += allocation[p][r];
                    printf("Process %d can finish.\n", p);
                    finish[p] = 1;
                    found = true;
                    count++;
                }
            }
        if (!found) {
            deadlock = true;
            break;
        }
    }
   if (deadlock) {
        printf("System is in a deadlock state.\n");
       printf("System is not in a deadlock state.\n");
   return 0;
}
```

```
Process 1 can finish.
Process 3 can finish.
Process 4 can finish.
Process 0 can finish.
Process 2 can finish.
System is not in a deadlock state.
```

```
A Sendled Detection
    Hirelade astdio. hy
    # Intlude & stdhoolin)
    # define P5
   #dafine R3
   in main Of . . . . . .
       int finish Ep3=404; .
         int wOAR [R];
         int need crains = 5
        £ 7, 5, 33;
          43,2,23,
        4 9,0,27,
          1 2, 2, 2 4,
          14,3,323;
        intellocation Erstel : 2
          £0,1,0),
           12,0,01,
           £ 3,0,2 k,
          {211,13
          10,0,23 3;
          irt available [R] = {3,3,23;
         For Cint 1 = 0; 128; 1++) &
word cis = avoitable Eis; 3
```

```
hool dudlock = fala;
            Int count = 0;
            While (count 2P) C
                hood found = fulk;
                  ror (int p=0; p < 1; p+){
                   18 (Fraish 203 =0) {
                    bool confinish = bur;
                     For (int r-0; PCR; r+1) {
                      if (need Epiter) -allocation Effert) win
                         Con Finish = false;
                          hecaki
                     4
                  if come Finish ) {
                    for lit +0; rep; +H) {
                      workers += allocation cpsses; 5
                   33
              if ( found) {
                    deadlock - true;
                    break;
              ; f(dead lock) & print PC4 State is indes 1 & & State W
                else of perintacions is not in deadlock station
                 return 0;
         Process I can finish
Output
         Process 3 car finish
        frocess 4 continish
         Process o confinish
       Process 2 can finish
        System is not in adeadlack state
```

Question

Write a C program to simulate the following contiguous memory allocation techniques a) Worst-fit

- d) Best-fit
- e) First-fit

=> Best fit, worst fit, first fit

```
#include <stdio.h>
struct Block {
    int block no;
    int block size;
    int is free;
};
struct File {
    int file no;
    int file size;
};
void bestFit(struct Block blocks[], int n blocks, struct File
files[], int n files) {
    printf("Memory Management Scheme - Best Fit\n");
printf("File no:\tFile size:\tBlock no:\tBlock size:\tFragment\n");
    for (int i = 0; i < n files; i++) {
        int best fit block = -1;
        int min fragment = 10000; // Initialize with a large value
        for (int j = 0; j < n blocks; j++) {
            if (blocks[j].is free && blocks[j].block size >=
files[i].file_size) {
                int fragment = blocks[j].block size -
files[i].file size;
                if (fragment < min fragment) {</pre>
                    min_fragment = fragment;
                    best fit block = j;
                 }
            }
        }
        if (best fit block !=-1) {
```

```
blocks[best fit block].is free = 0;
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
files[i].file_no, files[i].file size,
                   blocks[best fit block].block no,
blocks[best fit block].block size, min fragment);
    }
}
void firstFit(struct Block blocks[], int n blocks, struct File
files[], int n files) {
    printf("Memory Management Scheme - First Fit\n");
printf("File no:\tFile size:\tBlock no:\tBlock size:\tFragment\n");
    for (int i = 0; i < n files; i++) {
        int found = 0;
        for (int j = 0; j < n blocks; j++) {
            if (blocks[j].is_free && blocks[j].block_size >=
files[i].file size) {
                blocks[j].is free = 0;
                int fragment = blocks[j].block size -
files[i].file_size;
                printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
files[i].file no, files[i].file size,
                        blocks[j].block no, blocks[j].block size,
fragment);
                found = 1;
                break;
            }
        if (!found) {
            printf("No suitable block found for File %d\n",
files[i].file no);
    }
void worstFit(struct Block blocks[], int n blocks, struct File
files[], int n files) {
    printf("Memory Management Scheme - Worst Fit\n");
printf("File no:\tFile size:\tBlock no:\tBlock size:\tFragment\n");
    for (int i = 0; i < n files; i++) {
        int worst fit block = -1;
```

```
int max fragment = -1; // Initialize with a small value
        for (int j = 0; j < n blocks; j++) {
            if (blocks[j].is free && blocks[j].block size >=
files[i].file size) {
                int fragment = blocks[j].block size -
files[i].file_size;
                if (fragment > max fragment) {
                    max fragment = fragment;
                    worst fit block = j;
                 }
            }
        }
        if (worst fit block !=-1) {
            blocks[worst fit block].is free = 0;
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
files[i].file no, files[i].file size,
                   blocks[worst_fit_block].block_no,
blocks[worst fit block].block size, max fragment);
    }
}
int main() {
    int n blocks, n_files;
    printf("Enter the number of blocks: ");
    scanf("%d", &n_blocks);
    printf("Enter the number of files: ");
    scanf("%d", &n files);
    struct Block blocks[n blocks];
    for (int i = 0; i < n blocks; i++) {
        blocks[i].block no = i + 1;
        printf("Enter the size of block %d: ", i + 1);
        scanf("%d", &blocks[i].block size);
        blocks[i].is_free = 1;
    }
    struct File files[n files];
    for (int i = 0; i < n_files; i++) {</pre>
        files[i].file no = i + 1;
        printf("Enter the size of file %d: ", i + 1);
        scanf("%d", &files[i].file size);
    while(1) {
```

```
int choice;
    printf("Choose Memory Management Scheme:\n");
    printf("1. Best Fit\n");
    printf("2. First Fit\n");
    printf("3. Worst Fit\n");
    printf("[ANY KEY]. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    // Reset blocks for allocation scheme
    for (int i = 0; i < n blocks; i++) {</pre>
        blocks[i].is free = 1;
    }
    switch (choice) {
        case 1:
            bestFit(blocks, n blocks, files, n files);
            break;
        case 2:
            firstFit(blocks, n blocks, files, n files);
            break;
        case 3:
            worstFit(blocks, n_blocks, files, n_files);
        default:
        printf("Closing...");
            return 0;
    } }
    return 0;
}
Output:
```

```
Enter the number of blocks: 5
 Enter the number of files: 4
 Enter the size of block 1: 100
 Enter the size of block 2: 500
 Enter the size of block 3: 200
 Enter the size of block 4: 300
 Enter the size of block 5: 600
 Enter the size of file 1: 212
 Enter the size of file 2: 417
 Enter the size of file 3: 112
 Enter the size of file 4: 426
 Choose Memory Management Scheme:
 1. Best Fit
 2. First Fit
 3. Worst Fit
 [ANY KEY]. Exit
 Enter your choice: 1
 Memory Management Scheme - Best Fit
                File_size: Block_no:
 File no:
                                                Block size:
                                                                 Fragment
 1
                 212
                                                 300
                                4
                                                 500
                                                                 83
 2
                 417
                                 2
                                                                 88
                 112
                                                 200
                 426
                                                 600
                                                                 174
 Choose Memory Management Scheme:
 1. Best Fit
 2. First Fit
 3. Worst Fit
 [ANY KEY]. Exit
 Enter your choice: 2
 Memory Management Scheme - First Fit
 File_no:
                File_size:
                                Block_no:
                                                 Block_size:
                                                                 Fragment
 1
                 212
                                 2
                                                                 288
 2
                 417
                                                 600
                                                                 183
                 112
                                                 200
                                                                 88
 No suitable block found for File 4
 Choose Memory Management Scheme:
 1. Best Fit
 2. First Fit
 3. Worst Fit
 [ANY KEY]. Exit
 Enter your choice: 5
 Closing...
```

```
O want fet, but fit, find fit algorithms
```

```
# include estato, h)
define structure blak with block-bu, block
All the structure file with Alleno, file - size.
void not fit Come + Block Heck & I'm+ in - Norks
   File fices
  Street Block &
   int Hock-ro;
    int block-size;
    int is-free;
   Struck Files
  int file_no,
    int file-5-22,
    3,
   void birt Fit Estruct Blak blocks I just & blocks, second Net
                                 int n-Files) 1
   print + Cafile_no: 16 filesize: It Block_no: 1 Block Sur 165.
    for (inti= 0; 14 n- Files; 1+ of
          in+ bis+ - fit-block = -1;
           int pain Fragment = 100000;
        for (int ) = 0; j < n - Hocks ; j++ ) }
            if thockseil. is free & & blocks withheles
             files Ci3, file-size)}
          if (fragment & min-fragment) {
                min fragment = tragment;
```

if (higt _ f: b_ hlock ! = -19 &

bust- Fit - block = i



Hochs [hat _ Fit - block] . 15 - free = 0', pointf ("KUlth "dit tid this dit to de to the" Piles Eiz. File-no, Filestiz, file-size, blocks Chst_Tit - hlock J. blak -no , blex b SE bisite - b/ock3blacksize, min-tragmin); 33 void firstfit astruct Nock Harks [3, int n-Hocks, struct File files [], int n-files) { printfle First fit "); printf "in File-no: It File-size: It Black no: It Black -5120: 1 t mag ment 1 h"); For (int ;= 0; ; in- P.les; ++) { int found = 0; Forling; = 0; JK H- Hlaks; j++ 72 hlocascilistree = a: int fragment = blacks Ej 3 Alock - size - Tiles Eij. tilesus, printe ("bd 1+1+1.d to + r.d the rolyn", Filestiz File-no; Files Eiz file size; J 7. void woistfit Cstruct Block block El, int n-blocks) other ? point fl "herosy many course scheme - bosset to me"); print flatile no . It File size: (6 Block - No : 14 It/dus folios ci] . File -10, Blacki] , File- Size, blockscoren

434.

hlock Ivant + i 1 - block I block - size, more skyme)

```
Buthal : Jamely unanequar John :
        Enter the cumber of Machs: 5.
        Green the number of files . 4
        Erter the size of the Preb.
        Kleck I Inc
        Black 2 Soo
        Block 3: 200
        Ble 2 4: 300
        Block S: 000
        Enter the size of the files:
      f lu 1:212
     File 2: 417
     51,3-112
      F. 1c 4: 16 26
1. First 414
   2. Bibl Eit
   3 Werst Fit
    CANYKEY J. Eric
  EASER gove choice: 1
    Best fit
    File-to:
             File 5; 29 '-
                    Block-10: Block-5121:
             212 - -
                               300
             417
                                500
    3 -
             112
                               200
              426
    ·h
                      .5
                                600
```

Question

Write a C program to simulate page replacement algorithms a) FIFO

- d) LRU
- e) Optimal

=> LRU & Optimal

```
Code
```

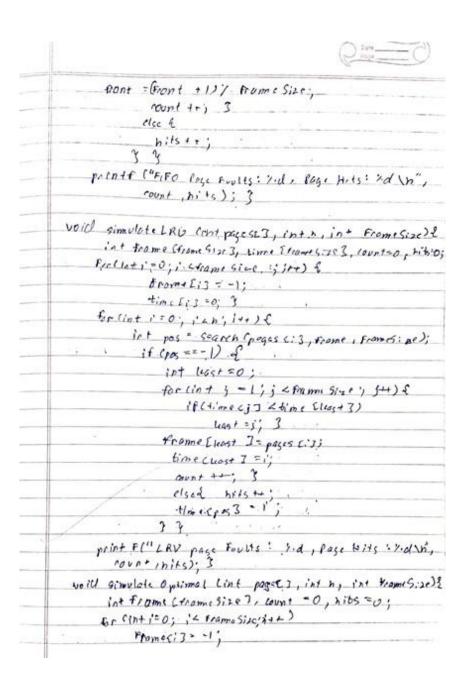
```
#include <stdio.h>
#include <stdlib.h>
int search(int key, int frame[], int frameSize) {
    for (int i = 0; i < frameSize; i++) {
        if (frame[i] == key)
            return i;
    }
    return -1;
}
int findOptimal(int pages[], int frame[], int n, int index, int
frameSize) {
    int farthest = index, pos = -1;
    for (int i = 0; i < frameSize; i++) {</pre>
        int j;
        for (j = index; j < n; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                     farthest = j;
                     pos = i;
                break;
            }
        if (j == n)
            return i;
    return (pos == -1) ? 0 : pos;
}
void simulateFIFO(int pages[], int n, int frameSize) {
    int frame[frameSize], front = 0, count = 0, hits = 0;
    for (int i = 0; i < frameSize; i++)
        frame[i] = -1;
```

```
for (int i = 0; i < n; i++) {
        if (search(pages[i], frame, frameSize) == -1) {
            frame[front] = pages[i];
            front = (front + 1) % frameSize;
            count++;
        } else {
            hits++;
        }
    printf("FIFO Page Faults: %d, Page Hits: %d\n", count, hits);
}
void simulateLRU(int pages[], int n, int frameSize) {
    int frame[frameSize], time[frameSize], count = 0, hits = 0;
    for (int i = 0; i < frameSize; i++) {</pre>
        frame[i] = -1;
        time[i] = 0;
    for (int i = 0; i < n; i++) {
        int pos = search(pages[i], frame, frameSize);
        if (pos == -1) {
            int least = 0;
            for (int j = 1; j < frameSize; j++) {
                if (time[j] < time[least])</pre>
                     least = j;
            frame[least] = pages[i];
            time[least] = i;
            count++;
        } else {
            hits++;
            time[pos] = i;
        }
    printf("LRU Page Faults: %d, Page Hits: %d\n", count, hits);
}
void simulateOptimal(int pages[], int n, int frameSize) {
    int frame[frameSize], count = 0, hits = 0;
    for (int i = 0; i < frameSize; i++)
        frame[i] = -1;
    for (int i = 0; i < n; i++) {
```

```
if (search(pages[i], frame, frameSize) == -1) {
            int index = -1;
            for (int j = 0; j < frameSize; j++) {
                 if (frame[j] == -1) {
                     index = j;
                     break;
                 }
            }
            if (index != -1) {
                frame[index] = pages[i];
            } else {
                int replaceIndex = findOptimal(pages, frame, n, i +
1, frameSize);
                frame[replaceIndex] = pages[i];
            }
            count++;
        } else {
            hits++;
    printf("Optimal Page Faults: %d, Page Hits: %d\n", count,
hits);
}
int main() {
    int n, frameSize;
    printf("Enter the size of the pages: ");
    scanf("%d", &n);
    int pages[n];
    printf("Enter the page strings: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &pages[i]);
    printf("Enter the no of page frames: ");
    scanf("%d", &frameSize);
    simulateFIFO(pages, n, frameSize);
    simulateOptimal(pages, n, frameSize);
    simulateLRU(pages, n, frameSize);
    return 0;
}
```

```
Enter the size of the pages: 7
Enter the page strings: 1 3 0 3 5 6 3
Enter the no of page frames: 3
FIFO Page Faults: 6, Page Hits: 1
Optimal Page Faults: 5, Page Hits: 2
LRU Page Faults: 5, Page Hits: 2
```

```
8 LRU, FIFO, Optimal Coprogram
   Hiralude < addio. h)
    #Include La Wlib . h >
     int search (Int hay , int Morreld , int formesien) ;
           For Cot i = 0, i & France Size ; in+ ) {
                if (from Eig=: My)
                   return 1; 5
        in + find Optimal Plat page [] , Int From (I), In ,
         int Inder , Int from Size ) (
             int fortnest = Indix, pos =- 1;
              forlint 1 = 0; i'c Frene Size; it ) {
               Intii
            for () = indet; jun, j++) &
                  If Cfrome E1] == pages .cj ]) f
                      if ( ) parthest) (
                      Erthest = 5;
                  breok; 33
             if(j== n)
                return 1; 1
              ratura (pos ==-1)? 0: pos; 7
    void simulate AFO (ent pages EZ, int b, int from Se
          int From e Expancisize I, Front = 0, count =0, +14
          for Cint 100; il Framesize jit+1)
                  Elorme I: 3= -5
           for (int i =0; i+x; i++) ?
                if a sweet a change Eiz , Frame, Promesials
                   Frame Chant ) = pagiscisi
```



Simulate 2 RV (page s, h, romes; 20);
returno;

bothet:

Enter the size of the pages:

Enter the moral page manual Strings:

1 3 0 3 5 6 3

Enter the ro of page frames:

3

FIFO lost Feets: 6, lage Hits: 1

Optimal loss foults: 5, lage Hits: 2

LRU lage Foults: 5, lage Vies: 2