# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



# LAB REPORT on

# **OPERATING SYSTEMS**

Submitted by

VISHWAJIT ANAND (1WA23CS046)

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 Vishwajit Anand (1WA23CS046), 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.	Experiment Title	Page No.
No.		
1.	Write a C program to simulate the following non-pre-emptive CPU scheduling algorithm to find turnaround time and waiting time.  →FCFS	1-17
	→ SJF (pre-emptive & Non-preemptive)	
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)	18-30
	→Round Robin (Experiment with different	
	quantum sizes for RR algorithm)	
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.	30-38
4.	Write a C program to simulate Real-Time CPU Scheduling algorithms:  a) Rate- Monotonic b) Earliest-deadline First c) Proportional scheduling	39-48
5.	Write a C program to simulate producer-consumer problem using semaphores	49-54
6.	Write a C program to simulate the concept of Dining Philosophers problem.	54-61
7.	Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.	62-67
8.	Write a C program to simulate deadlock detection	68-71
9.	Write a C program to simulate the following contiguous memory allocation techniques a) Worst-fit b) Best-fit c) First-fit	72-83

10.	Write a C program to simulate page replacement algorithms a) FIFO	84-92
	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.
	Conduct practical experiments to implement the functionalities of
C04	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.

```
\rightarrowFCFS
```

→ 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\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:

```
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
Process 2: 0
Process 3: 0
Process 4: 0
Process Arrival Burst Waiting Turnaround
        A
                                                   10
                                  14
20
                         10
                                                   14
        a
                                                   20
        0
Average Waiting Time: 7.75
Average Turnaround 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++)</pre>
```

```
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
        0
                7
                                7
                                        0
1
                                                 0
                                6
                                        3
        8
                        14
3
        3
                4
                        11
                                8
                                        4
                                                 4
        5
                                        9
                                                 9
4
                6
                        20
                                15
```

	Comment
1	) first come First series -
	Server -
	# include <s dio.h=""></s>
	Hindude Cranio. h>
	thinducte ( muth, h)
-	struct process &
-	int process no
+	int arrival to
4	Mut responds is
-	Int busyt time;
-	And Color Mat as
1	int waiting time;
+	aint turn - around time;
1	
1	Void mitt
1	void sort (struct process als) in h)
1	inth) ?
1	unih) s
_	for (juti=0; i < n-1; i++) ?
1	700 1 467 170
	for (int j=0; j <h-j-1; ((a1[i].arrival-time)="" if="" j++);=""></h-j-1;>
	(Att ?
	(at GJ. arrival -tin)) 9
	Struct process but = asci
	مارين = سارين:
53	3 allij ] = tempi
	3
	3

	Date
1.	
	void fifs (struct process als); in his
	int current time =0;
	for lind iso; ichi i+1) s
	currents tim = fmax (current time
	alli). Vespons -tim = curry-tie
	all avriva - the
	werent time += cascis. buse the
	alli]. completion time -
	current time;
	ascil torn-around tim =
	ascis, composión timo-
	astis. arrival time;
	Will waiting time = ascil.
	turn around tibre -
	allil.arrival -tim;
	3
	void proposo ( Etract process as []
177	int n)c
	for (int i=0; i(h; i+4) {
	printf & Enter Process now
	arrived time, burst time
	Stong (" 1.0" , as [i] , process
	sing (" Y.d", & willia ] arrival
	sant (" 7.d", Lascis. burst ti)
	3

	int main () ?
	man () (
	int n= 109;
	Sand ( process proc Bh]:
	Sort ( proc + n);
	projects ( proc. )
	prints (proc. h);
	3
V	printf (" In PID VAT ) + DIL i ant n) ?
	print+ (" In PID It AT ) DE wint h) &
	for (ibd i=0; i L n; i+1) ?
	Print ("No 21)
	print ("/h x d / t x d
	10 1 1 1 0 1 t 1 d
	ast. 17. process id, astil. arrival
	tine, alli] burst-tin,
-	all ide response time?
	ascid. completion time,
	alrize with the
	astidium aromentia);
	1
/	3
	****

_	METRO Per
_	De dela
_	011(
_	Odf (min-index ==-1) ?
_	3 CULTEN - + 4-1:
_	els 9
	ps [min_indox] start -t = current
	tomen indox I start - 1 - 1
	pscmin in a
	ps [ mih_index ], but = as [ mil_index]
	ps [min_index]. fot = ps [vnih_inder  ct - ps [min_index]
	total TAT += ps [min indox ] tot;
	ps[min_indox]. wt = ps[min_indox]
	total WT + = ps[min_index]. wt;
	V) sited [min-index 3: +ruz;
	completed +1;
_	current = ps [min - index ].ct;
_	)
	3
_	prima of (" The away TAT is : 1.2 f ms",
_	Posto Stotal TAT In );
	printf("The average ho isig. of ms",
	tow MI (N)
	N. L.
	3

_	METRO Per
_	De dela
_	011(
_	Odf (min-index ==-1) ?
_	3 CULTEN - + 4-1:
_	els 9
	ps [min_indox] start -t = current
	tomen indox I start - 1 - 1
	pscmin in a
	ps [ mih_index ], but = as [ mil_index]
	ps [min_index]. fot = ps [vnih_inder  ct - ps [min_index]
	total TAT += ps [min indox ] tot;
	ps[min_indox]. wt = ps[min_indox]
	total WT + = ps[min_index]. wt;
	V) sited [min-index 3: +ruz;
	completed +1;
_	current = ps [min - index ].ct;
_	)
	3
_	prima of (" The away TAT is : 1.2 f ms",
_	Posto Stotal TAT In );
	printf("The average ho isig. of ms",
	tow MI (N)
	N. L.
	3

-	NETTED Page 7
0	If (min-inger ==-1) &
	3 Current -+ ++;
_	else 9
	ps [min_index] start -t = current
	start + 1 - PS with inde
	total TAT += ps [min index] at
	tet - ps[min-judos
+	15/mg-1ndox 111
	V) sited [min-index 3: + ruz;
-	current = ps[min_index ].ct;
	3
	prima f (" The average TAT is: 1/2f ms", token total TAT /n);
	printf("The average hor isig. 2f ms")
	total WT /h);
	3

D	Shortest Job First (preemptive)
-	Direction of the Control of the Cont
-	the include a statio h
	the windled & stollbook h>
	BY CINCIDIO O BY CONTROL S
	struct process s &
	int pict.
	int at, bt, cf, wt, tot, start tiz;
	3 PS[100];
	N. A. C.
u	nt main () 9
_	int hi
_	flout bet remaining [100];
_	bool is completed [100] = 8 fax 3:
_	int current -th =0, completed =0;
	glout . Sum - tot = 0, Sun wit = 0
	3
	print f (" Enter no. of processes");
	Swe f (104.0)1, & n);
	Pay (inhi =0, ix bi it) &
_	
	print to
_	Printf(" Enter arrival times");
,	for (Int vi=0; i < h; i+1) ?
	postat for
	Scart ("Y.d", & ps[i].at);
	ps[i].pid=i;
_	- 3

print (" Ender Burst Time").
for (int :=0; ich : 411)
conf (" 1, d" , & pot 17 1, )
by variating [i] = ps[i] by
3
while (rompleted 1=h) ?
ind smih_index = 1;
int min = 99999;
for ( int i=0; i(h; i+1) &
It ( pstill at <=
current_tim &&
Lis confiden [i] 22
by verneining < min)
Smin = bt - remain [i
mih_indox = ii;
3
3
uf (min-index == -3)9 current-fin ++) 3
current_fin ++) }
2152 9
else 9 if (bt - verning [min_ibled== ps [min_ibde*], kt ] 1
ps [min index ], bt ] 1
 ps [min index], sant
= Current time;
3
ht - Vamoing [min-indox]-
prov= current the;
•

```
if (bt romainy [min_ingles == 0) &

PS [min_index] totat == (correct tt;

QS [min_index] tot == ps [min_index] correct tt;

PS [min_index] tot == ps [min_index] tot;

Sun_tot += ps [min_index] tot;

Sun_tot += ps [min_index] tot;

completed tt;

is completed tt;

is completed [min_index] tot;

tku;

print (" [n Akorogo tot; "/f",

Sun_tot [n];

print (" [n Akorogo tot; "/f",

Sun_wt [n];

votorn 0;
```

```
#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] <= 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
                                TAT
                BT
                        CT
                                        WT
                                                RT
        0
                8
                        17
                                17
                                        9
                                                9
1
2
        1
                4
                        5
                                4
                                        0
                                                0
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) {</pre>
        min priority = 9999;
        highest priority = -1;
        for (int i = 0; i < n; i++) {
            if (at[i] \le time \&\& flag[i] == 0 \&\& pr[i] < 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++) {
       printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[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
                        Priority
                                                                 RT
                BT
                                         CT
                                                 TAT
                                                         WT
                                                 14
                                                                 10
1
        0
                4
                        2
                                         14
                                                         10
2
        0
                10
                        1
                                         10
                                                 10
                                                         0
                                                                 0
3
        0
                3
                                         17
                                                 17
                                                         14
                                                                 14
        0
                        4
                12
                                         29
                                                 29
                                                         17
                                                                 17
```

## => Priority Scheduling (Preemptive):

```
#include <stdio.h>
struct Process {
    int id, arrivalTime, burstTime, remainingTime, priority;
    int waitingTime, turnaroundTime, completionTime;
};
int findHighestPriority(struct Process p[], int n, int currentTime) {
    int highest = -1;
    int highestPriority = 1e9;
    for (int i = 0; i < n; i++) {
        if (p[i].arrivalTime <= currentTime && p[i].remainingTime > 0)
{
            if (p[i].priority < highestPriority) {</pre>
                highestPriority = p[i].priority;
                highest = i;
            }
        }
    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\t
Waiting\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].id,
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
8
9
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 3 16 16 8
4 0 2 4 18 18 18 16

Average Waiting Time: 6.75

Average Waiting Time: 6.75

Average Waiting Time: 6.75

Average Turnaround Time: 11.25
```

,	) Priority sequencing - Precontin
9	# windledo <setdio.h></setdio.h>
	# induct 2 stallb. h>
	TI THANKS CHAISIN
	typing structos
	int mid i
	int at, bt, priority, ct, tut, wa
	( Constally)
	3 procos [ 100];
1	will precuptive priority ( 1000)
1	MY 3.
V	oid preempix prioriesty ( inth
	floor to Tat a the
	flowt * any TaT, flout * any NT )?
	int contacts
	int competed = 0, tim=6;
	int min-index;
	il total TA =0;
	int dotal NT = 6;
	unt is Conflicted [ n 7 8 = 803;
	, , , ,
	In I
	for ( unt 4=0; 11 /2; 211)
	dw 9=0; 11/1, 51/16
	for (int 1=0; 1 Lh; i++) {  \$00 p[i] vomining = p[i].  borse;

	NETRO
	Date: / /
	while (completed ( h) ?
	Completed ( ) C
	This - Indix = -1:
	min-Indix = -1; int min-priority = 5999; for (ind 2 = 0; )
	100 C 410 7 = 0 : 17 : 3993.
	15(0)
	if ( of in
-	P[J]. Frma Nis
_	PLJ - Prievis
_	P[#] , priority == min-priority
-	Bl p[i] .at asca <
_	PErmin-index J. at )) 5
_	min-priority = prij , pour
_	min- undo k = ij
_	3 mark = 4,
	3
-	3, 110 100 100
	if (military == -1) ?
	timett;
	contine;
	3
	p [min-under]. remaining;
	tim++;
	if ( PErnin-indox ). remains == 0 8
	p [ umin index ] ct = tim;
	PC win-ind or ]. Vat = p[inja. id.]
- 1	ct-ptinin_syler.u.,
	PEmin-inlay 2. n+ = prunin-indi
	tot - ptmin-1-void he
	is compreded [ min - index ] = 2;

Date:
total NT += p[min jndox]. tat  total NT += p[min jndox] n
Conflety ++ j.
7
* avy TAT = (flowid) total TAT /n;  * avy NT = (flowid) total hot/n;
ary mT = (flower) total ht/4;
3

Non Pro emprik Priority Schoduly # undude Letdio. h> tydes strut & int pideat, lot, It, remaining fin tot tat, wit, time, is comple 3 Process; non presomptile priority ( Prouss pl void int domprisity = 9999, geletel= for ( int i=0; i < h; i++) ? at a c= time ed 1 p[i] is completed el ptil. pt 2 lowpriority dow priority = p[i]. selocted =i if (state = - 1) ? time +11; Contino: time += p [ solder ] p [ solection ] ot = w p [ south ] . tat = p [ solder ] probability J. wt = production to 1 P [ Selvety ].

#### => Round Robin:

#### Code

}

```
#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++) {
            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\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
Process Burst Time
                        Waiting Time
                                        Turnaround Time
        10
                        15
1
2
                        15
        5
                                         20
3
        7
                        16
                                         23
4
        3
                        12
                                         15
Average Waiting Time: 14.50
Average Turnaround Time: 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];
            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
0
Enter Burst Time, Arrival Time and Queue of P2: 1
2
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
1
        0
                        2
2
                        3
3
        3
                        8
                                                3
4
        8
                        11
                                                8
Average Waiting Time: 3.25
Average Turn Around Time: 6.00
Average Response Time: 3.25
Throughput: 0.36
```

				-	Date	4 100	
V	old n	multilor	el mus	14			
_	Sych	em q	mane,	rity scho	Helis +		
	115 - 1	and the second second		The second secon	As to		и,
-	COO	DOOD,	10000	50000	OKAL	din !	BB,
				D DD CO	0.00.0	D at [	17,1
		the state of the s					
	vv(	Sucta	r-grow	- MEA-	(unua)		
	-	100	A CONTRACT OF	1 243 -	com,	LIME	
	Lies	CALA	Listi	n);		-	
_	1112	050	r- quo	e , wor	-10 m	. 14	1.
					The second second	A DIV	
	-						-
3							
				1 - 21			
				- 537			
0/P							
0/P					, [n]		
l in	-)	AT			i ini		
0/P	-)		ßT	Eure	īu.	7AT	R°
l in	-)	AT		Ewe	i ini	TAT	R.
Proces	-)	A T			nT 6		R.
Proces	-)	A F	8T 2 1	Ewe	wT	TAT	R.
Proces Pl	-)	A F	BT 2	Ewe	nT 6	7AT 2 7	
P1 P2 P3	-)	A Too	8T 2 1	<u>1</u> 2	νī δ 2	7AT 2 7 8	R.
P1 P2 P3 P4	3	A Too	8T 2 1 5	<u>1</u> 2	νī δ 2	7AT 2 7 8	R.
Ploces Pl P2 P3 P4 Avg	s w1	A 1000	3T 2 1 5 3	<u>1</u> 2	νī δ 2	7AT 2 7 8	R.
P1 P2 P3 P4	s w1	A For	6T 2 1 5 3	<u>1</u> 2	νī δ 2	7AT 2 7 8	R.

Multi Lovel Quene Schodulinga # dofine MAY-ProcESSB 16 # define TIME\_QUANTUM 2 type of struct & int at, by queux -type, hed, text, vt, rem\_tin; 3 Process' void vr (process processes [], whith, jut time-quantum, int to dia int done, i;

	Prom 79
Y	do 8
Y	
Y	don = 1;
Y	for ( i=0; i(n; J+1) ?
Y	if (processati). on rom_HA 70
Y_	aon =0;
	if Cprocesses [i] · Vemati
	tim quantar) ;
	+ time += time-qua
	processes til 7. rom.
	-= tin-gum
	70128
	+ tim+ += processes E
	avaluation avaluation of the state of the st
	(ACCESSION 7 OF 2 )
	-processes[i].6
	processes Cil, b
	proposses [i] tet=#tine-
	process Ci7. at
	pvocessos City yt =
	processes [i] wt
	processos Ci7, vom - +1n =0;
	3
	J
	3 while (1 dons);
	3
-	
-	

void -	11ts ( process processes [], july,
	JUL 1111 C
for	(ind is; ich; ith) 5
	* aim = proprocesses [i].
	3
	process estil. wt = + din - process
	- DV0(03303 M 1 . 1. 1
	process castion by
	+ vime += processes [3] - 141;
	2
-	
00000	Sout ( User-que Dente Service Service)
VOICE	THE PROPERTY OF THE PROPERTY OF THE PARTY OF
for	Sinti= 6; il mor cont
TOV	JUNIA- 6, AL JUSOV COUNT - 1 1 1111
TOV	for ( what j=0; j < user_round -j-1;
TOV	for ( ihtj=0; j < user_10md-j-1;
TOV	for ( what i = 0; i < user_round - i - 1; i++);  if ( user - quanti). at 7
TOV	for ( ihtj=0; j < user_10md-j-1;
TOV	if ( ush - quan [j]. at ?  ush - quan [j]. at?  ush - quan [j+1]. at ) {
TOV	if ( ush - quan [j7. ot 7  ush - quan [j7. ot 7  ush - quan [j1]. ot 7  ush - quan [j+1]. ot ) {  Process clamb = vor - quan [j1]
TOV	if ( ush - quan [j7. ot 7  ush - quan [j7. ot 7  ush - quan [j1]. ot 7  ush - quan [j+1]. ot ) {  Process clamb = vor - quan [j1]
TOV	if ( ush - quan [j]. at ?  ush - quan [j]. at?  ush - quan [j+1]. at ) {
TOV	if ( ush - quan [j7. ot 7  ush - quan [j7. ot 7  ush - quan [j1]. ot 7  ush - quan [j+1]. ot ) {  Process clamb = vor - quan [j1]
TOV	if ( ush - quan [j7. ot 7  ush - quan [j7. ot 7  ush - quan [j1]. ot 7  ush - quan [j+1]. ot ) {  Process clamb = vor - quan [j1]

		-		Date	1	
Void	but hit				-	-
S.	multiles stem q	el prio	rity sch	steluti-	-	_
1.1	Date Barrier Brown Co.			As to		
- 0	50x-10up	TODGO	R Day	won, 4	tin !	CENC
	200000	11	2000	d part	D at 1	7
2	ort (uso	r-quew	usa			
Y	19.00	A CONTRACT OF	1 243 4			
	QUAN	Lighti	m);	· com,	LIME_	
-(1	es ( uso	V- 942				
		VICE I	e just	-104m	Stir	1:
5						
3	7-11					
3			- VI			
3						
0/19	1 2 2			i fal		
0/1-2	AT_			i fall		
3	A T bot	ßT	Quant	, NĪ	TAT	R.
0/P-) Process	POL	BT			TAT	R.
0/P-) Process	O -	BT 2	Quant 1	٥	TAT 2	
0/P-) Process P1 P2	POL	2		٥		
0/P-) Process P1 P2 P3	0	2 5	2	2 . 7		R. 0
0/P-) Process P1 P2	0	2	2	٥	2	
0/P-) Process P1 P2 P3	0	2 5	2	2 . 7	2 7 8	
0/P-) Process P1 P2 P3 P9	0	5 3	2	2 . 7	2 7 8	
O/P-) Process P1 P2 P3 P4 Avg	0 0 0 0	2 5	2	2 . 7	2 7 8	
O/P-) Process P1 P2 P3 P9 Avg T	0 0 0 0	2 1 5 3 .25	2	2 . 7	2 7 8	

#### 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++)
        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++)</pre>
        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].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++) {
                 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;
}
```

```
Output:

Time Ta
0 Ex
1 Ex
2 3
4 5 Ex
6 Ex
7 8
9 10 Ex
11 Ex
              Task 2 Task 3 Task 1
Exec
              Exec
                           Exec
                           Exec
                                         Exec
              Exec
              Exec
                                         Exec
                                         Exec
              Exec
11
12
13
14
15
16
17
18
              Exec
              Exec
              Exec
19
```

S	Pale Monotonic -
	H= indudo (ddio. h)
	typody servet &
	ind spriod;
	int burd;
	int comi
	3 Task;
	int god ( ind a, intb) ?
	return b==0? a: 90d (b: 11)
	3
	int Jem (inta, ind b)?
	restorn at b/gca (a,b);
	3
	int find typer priced (Task tusks[] unt
-	for ( and i=1; i < h; i+1) ?
	hypr= Jem I chyper, stushotis.
	Poriod);
	Exturn chyps:

	23
	void cut
_	for (int i= 0; ich; it) o
	for fi
	to a contrict.
	for (int i= 0; ich i it) ?
_	J-00-41:
	for (int jeto it 1 ijchijt)?
	itushed Cineman
	pariso ) ? 5
-	Test
	Task tomp = tasks[i]:
	stacks[; 7 - Lasks[i].
	Talbert. 7
-	3 to the
	3
	Lakan 12
	tashs[i]. rem= tashs[i]. bors)
	3
	3
	20
V	oid rate Monodonic ( Task tushs [], in h
	int sim time) ?
-	
	- 1040 010 1
	printf (" In Pate - Monotonic School
	prind [ [ Task ) + Task 16");
	sort By Porled (tusks, h);
	for ( int time = 0; time L sim_time
	time ++) 8
	ant shalulad = -1;
I	for (int i=0; i (n; u++) ?
	for (unt y=0, U ch, with )
Ta:	1.7.7
1,	if ( tim / tosks [i] por tasks[i], rem = tasks [

```
for ( und i= 0; i (h; bi++) ?
          if (toshs[i] . remaining 20)
                Schodulod: ij
                 bushi
        if (schoduba 1= -1) 5
            to sto [ cateduted) very --
            print ("1. d 1 + 7 1. d 1 n"
                   tim, tustes Ischodulas
        olse 8 printf("Y.d ) & Table 16"
                             time );
int main () {
   int hi
    printf (" Entor noc, of
    Scort (117. d', En)
    TO COMO
    Task tasks[h]
     for (int i=0; i 4 h; i++) 8"
         tocks[i] pid = i+1;
          printf (" Ender hound and butst
             time for task Ty. d:", it 1)
         Scort (14. d 7. d", & tout 5 Ci]. ported
          & dashs [i) burst );
```

```
int sim_time = find Hyper Poriod (tasks, h)

print f (" Simulation will kum for 3.d

units (LCM of keriods) h",

Sim_tim);

Vate Monotic (tasks, book h, clime);

ktorno;

3
```

# => Earliest Deadline First

```
#include <stdio.h>
#include <stdib.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++) {
            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; <math>i++) {
            if (current time < tasks[i].deadline && tasks[i].executed</pre>
< tasks[i].execution time) {
                executed task = i;
                tasks[i].executed++;
                break;
            }
        }
        if (executed task !=-1) {
            for (int i = 0; i < num tasks; <math>i++) {
                if (i == executed task) {
                    printf("Exec\t");
                 } else {
                    printf("\t");
            }
        } else {
            for (int i = 0; i < num tasks; i++) {
                printf("\t");
            }
        printf("\n");
    }
}
```

Output:

```
Time
         Task 1 Task 2 Task 3
0
         Exec
1
         Exec
2 3 4 5 6 7
                  Exec
                  Exec
                           Exec
                  Exec
         Exec
                           Exec
8
                           Exec
9
10
         Exec
11
         Exec
12
                           Exec
13
                  Exec
14
15
                           Exec
16
                  Exec
17
18
19
```

# Program 5

## Question

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

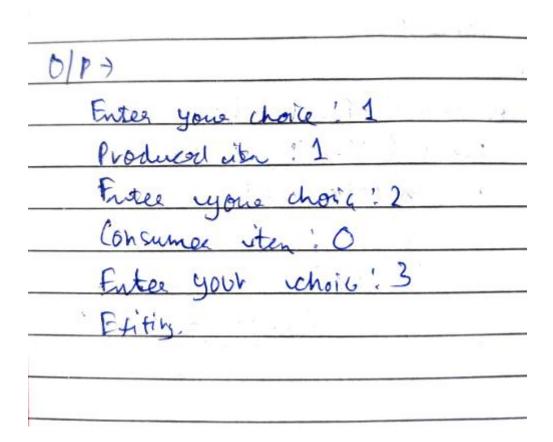
#### => Producer Consumer

```
#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);
    } else {
        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
```

20	Product Consumor
	# include < St dio.h>
-	int muty = 1, full = 6, u= 6, samply = 2
1	Joseph ( (+ S)); 3
	int signal (int #5) worn (++ (#5)
1	void producor () ?
	mait (& uniter);
	signed (& full);
	wait (& ompty);
	utt;
	priktf la Item produced so ya sa
	Signal (& mutex):
	3
V	id consumor () ?
	wait (& mid (x);
	wait (e zul);
	signal (dompte):
	print (10 Can Suma with m 2d/h"
3	signal (& mentex);

	Name of the Control o
_	Propriet
	2-4-
_	int min () &
	int choice;
	do E
	print re
	Sconf (1'y.d" & chala);
	Switch (choic) ? chala);
	(95.1:
	if ( mudex == 1) 88
	(only)
	(12mpty!=0)) ?
	product (); 3
	to else print f (" Buffer is fue
	Bufter (State
	bown;
	(as ):
	1 f (mutex ==1) 8.8
	(full 1=6)) ?
	Consumer(); 3
	ols printf(" Buffer is ont;
	pkub;
	2.
	prinafi" Exiting po In"):
	a hear!
	while (choice 1=3);
	NAME TO SECOND
1	retorh O;



## Question

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

# => Dining Philosophers

```
//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 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
```

	Dining Philosphot
9	
	Hi Include ( pthread . b > 12
	It indude Comphasein
	H include Liddio. h.
	4 400 N S
	H I I THINKING
	# dollie HUNGRY
	U IN PATTING O
	+ deline LEFT (phrum + 7)/. N
	+ whin RIGHT (phonum +1)
	int state [N];
	int while [N] = 80, 1,2, 3,43;
	som _ t mutor;
	Sem_t SINJ;
	int main () ?
	int ii
	pothroug + throad id [N];
	som_init (dimuter, 0, 1);
	for (i=G;i(N;i+t)
	30m-ikt (2 S[i] ],0,0);
	for (x=6; j < N; itt) ?
	pthread - greate (8 thread - id I is
	NULL, philosper, applilio
	printf 1" Philosphor Y.d is
	3 thinking 1h", it 1);
	,

	27
	for (i=0; i/N; i++)
	10 = 0; v/ A;
	pthream in vita)
	3 agan (Hurani
	NIN Cid Oliver
_	Pthreunjain (thream is [:] NULL):  Void tost (und phonen) s
	State [ Phone ] == HUNGADRY 28  State [ KIGHT ] 1= FATING 28  State [ KIGHT ] 12 FATING 28  State [ Phone ]
	State Iphon ?
	State [1FFT] == HUNGANON
	State There FATTING
	12 5000 88
	(tot I)?
	pham ] = Ear
	State [phain ] - FATING:
	print (opilla)
	print ( o philosopher . I. d tob
	phonoman 1 1 FFT 11
	phrands, LEFT+1,
	phone (11)
	The things
	ENTING VIII
	Som post (& S [phonen ]);
	3
	1
	void tube - fork (in to)
	To the phan C
	Sem - Wait (& muteur)
_	State Ephner ) = HUNGEY;
_	privat (" Philosophu Y.d is Hungay 14"
	phonests);
	tost (phono);
	Com-Port (l unatex);
_	sem - West (& S[phnum];
	sleeh (1)i
	3

	Company of the Compan
	total and and and and
	void put for K ( int phrum) 5 cem-wait ( & muton);
Ī	A1 . F .
	State [ phoun ] = THINKING.
	prints ("Philosphee Y.d pulled fork oy.d and y.d
	do as il
1	Dhamada Isa
	primat(" Philosphox : 40 vis this
1	prina(1" Philosphor /d vis thinks
	tot (TABL);
	tour (RIGHT);
L	Som-post (& muter);
	3
-	
	void # philosphor (void # num) g
-	while (1) {
-	int * i = hun;
L	sleep (1);
_	take fork (+ i);
_	See (0);
_	Pret - tork (7 i);
	3
_	
	}
	3
	}
	}

				-		
0/10						
Phil	6SADHUN					
	11	o is	thinkin			
		1	11	4.		
	1.1	2	11			
	1/	2				
			finish	1 12 65		
	4.			- June	y open	Put
	211	1	1	don	tork	
	11	2		ween.		
			45	think	16	
		1				
		1	- /			
				200	11-7	
			TANK A	1 1		
		F	Part Francisco			

#### 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++)
        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;
Output:
```

Output.

```
Enter number of processes and resources (n x m order): 5 3
Enter Allocation Matrix:
010
200
302
211
002
Enter Max Matrix:
753
322
902
222
433
Enter Available matrix:
System is in a Safe State.
Safe Sequence: P1
                            P4
                                   PØ
                                          P2
```

```
on Damber is Algorichis
      Hindlude Ladions
      # include acidlining
       int condition Cent ** need, int our int , int , int on)
             for (in+ j = 0; j = m ; j =+)
                 if (need I SEI] > work Ej 3)
                   Atturn 0; 3
                return 1; 3
     in+ Satisty eint my into , in+ " advalled job + * maxisting
                                                     int squared
            in + to need = (in+++) neello (in+ size of (in+ ))
             for linti = 0', jan; jrt)
                mend 5 i3 = lint ( ) malbe (m + girl of (lot)).
                  For (int 1=0, 140) H)
                    herd Liz = (int *) malloc (me size of (int);
                     For (int j=0; j <m; j++)
                       heed Lis 23 1 = morsing is -allocated sight
        int sale irdex = 0;
        int changed;
            charged = 0
             for Mot i " O, ich ; jer )
```

```
if Clfinish cis & an condition (need, work, i, m))
  For Cint j=0; j 1 mo; j++)
         workesit= allocated EilEsl; ?
        Anish Ei 3 =1 :
         Sigurne Isofe Indez ++ ] = i;
        Changed = 1; 4 3
  I while (changed);
   for (int 1=0; 14n; 4+)
     if ( hnish [ is) f
      returno; 1 1 returnit; 7
ind main()
   int non;
   printf ("Enter number of processes and resources (nem)
                                                 order):"
   Scort (" V.d", Vn);
    Sonf( 1d", (m);
    printf ("Enter Allocation Madeix: \n");
     int + alterely = (nt ++ ) miglior (n' sin of (int ));
      for ( int i= 0; ikh; ith) &
       allocated Lis = Got + ) malloc continuos (int);
          for (in+ ; = 0; j < m; j++)?
            scanf covid", Fallocuted pil cj3); 33
      printf ["Enter Max Martin : Un");
       int + max = (in+ +) maller (n+ size of (in++));
         more Eiz = Get 1 malloc ca " suroffict") );
          For Cint i= Dj i Ln; in)
          1 marcis = (int ) mall oc ( in " sicrof cint y);
```

```
The Analelis motion

3 3 2

System is in Sole State

Sole signification of 1
```

#### 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;
}
```

# Output:

```
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.
```

```
& Scholleck Ditection
    Hinclude astdio. h)
    # Include ( stdhool. h)
    # define P5
   #diffine R3
    in main of . . . .
        int finish sp3=403; .
         int WOAR [R];
         int need crater = 5
        £ 7, 6, 33;
          43,2,23,
         4 9,0,27,
           12,2,27,
          1 4,3,32 35
         Intallocation Erstes ? ?
          £0,1,01,
           12,0,01,
           £ 3,0,2 k
           (211,13
           10,0,23 3;
         irt available [R] = {3,3,2};

For C:n+ i = 0; i = R; i++) f
              word cis = quoilable cis; 3
```

```
hool deadlock fala;
     Int count = 0;
     While (count 2P) &
        hool found = fulk;
          For clut P=0; pce; p+x
           18 (Flaish 1 =3 =0) {
            bool confinish = true;
             for (int r-0; rek; r+1) {
             if (need spice) -allocation friends wan
                 Con Finish = false;
                  hecaki
             4
           if (cap Finish) {
             for lint r= 0; r<r; r+1) {
              worker 3 += allocation cpssrs; }
           33
       if ( fourd) {
            deadlock : true;
             break;
      if (dead lock) & print PC" Action is indeplox & state.
        else & printf("System is not in deadlock statio
         return 0;
 Process I can finish
 Process 3 can finish
frocess 4 continish
Process O con finish
Process 2 can finish
System is not in adeadlack state
```

```
Program 9
```

#### 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++) {</pre>
        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\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:
 File no:
                               Block no:
                                                 Block size:
                                                                 Fragment
                 212
                                                 300
                                                                  88
 1
                                 4
                                                 500
                                                                  83
                 417
                 112
                                                 200
                                                                  88
                 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
                                                                  288
 1
                 212
                                                 500
                                                                  183
 2
                 417
                                                 600
                 112
                                                                  88
                                                 200
 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...
```

	Page 100 Date:
1	D) Bost Fit
	void bost tit ( struct Block blacks)
	and harrocks,
	ctuet till filest ]
	due no filos) s
	for list jeo; I (h-fils; il+) ?
	une bost fit black >-1;
	Just min-fragm2 = 10000;
	for land y=6; y lh-blocks;
	if (blocks [j], is-tree as
	blocks [j] . black - 522)=
	01:000 [4 1 : 01de 2 2 2 2 2
	filos [i] file size)s
	if (-fragment & hish -troogn
	min - froguera = frage
	bost fit block = j;
	3
	)
	3
	if (bost fix block 1=-1) {
	Kock & [ bet fil - block ]. 13 offer
	print fill \$70 (at y. id (at y. id)
	print file and City
	gilotal filener
	giles Ci J. file size
	Blocks F boxt - fit - bl
	block re
	hlocks [hosa, fix hos
	bloch Biz,
10	· min-fragment);

				and of	_
/	vol:	v 00:11	710 -4		
/		grines	(" 7.0 \)	1.014	
/		NOT Alle	etala file	stal fib	Sabs.
/			fil	elil fib	- Car
		3		-	
	- 3	-			
	-				
	0/87				
	Filo-he	Fib-siz	Block - ho	Black -siz	Floge.
_	1	212	3	300	83
		4 - 2			
	2	146	4	200	51
	3	426	2	560	74
_	9	464	N/A	4/4	w/I
		10 7		10/1	
			-		
		2 2			
	-				

11	Pirst Fit
	# include ( Staio. h>
	Struct Block &
	ind black -roj
	int block-size;
	int is-(kee)
	struct File &
	in file-no;
	ina file - stze;
	3
	void first Fit ( struct Block blocks [],
	Jul 1- 016U25.
	struct File file [2)
-	ile n-fills
	printf (a In Proposed First Fit ");
_	DW 08/37/2 30 X
	for (int i=0; i'L n-fibs; i++) ?
_	int allowed=0; &
-	for ( and jeo; j & n-block s; jet) s
-	If Poloche [ ]. is afken le
-	blocks[j]. block_siz>>
	files lij ]. file = liz) s
	in fraguer = hechstjo.
_	bloth -ziz - files [i].
	blowhs EjJ. is-francoj

print [10] . d (+ 1/0) + 1/d ld.
This Late That he was
files[i]. file .co
blocks[j]. block-ho
blocks[j]. block size
Commission Size
fragnet);
allowed = 1;
bkent',
- 5
 /
if ( ) allocated ) ?
 printffil y.d   + 1.d   + Not
 Alborata Lu'i,
filos [is ]. file-ha,
file Find
files [i] files -six);
3
3

_	Worst fix
127	Worst the
17	void moonest fit ( sixuet Black blocks)
-	void moonst fit int in blocks,
-	severa the files ?
1	in hafirs) &
+	JW hoffing
1	for (int i to i) < n-tibs; i++)5
	lov fint i did & n-tres
1	
1	man - fray "Co
+	for ( ind j=6; ij Lh-block; j++)
+	of (blacketi) a) s free de
-	blachs[j]. bloch -size >=
	files(i). file Siz) s
	files (1). The Top
	il hat fragman = bloch s[j] . bl
_	July Traymon = Diourse = . Di
	- files [i] file
-	if ( grament > man fraguer man-fraguers = fraguers
_	man-fragment = fragmen
	horst fit block = j
	1
	9
	1
	black of block 1=-1) &
-	block of movel oil
_	place & F movest fix - block 7. 15-f
_	1 1 1 1 1 1 1 1
	files [i] file no
	files [5], file size
	file co

15		
_	block I wort fit block I block -ro,	
	blocks [moret - fit - block ]. block - 51 &	,
	man fragmen);	
	3 olse E	
	print + ("Y.a \+Y. a \+ Not Alcounty	١٩"
	files [i] file _no,	
	filos(i); filo-siz);	
	3	
1	3	
	3	
1		

## Program 10

## 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++) {
        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++)</pre>
        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++) {
        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])
                    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;
}
```

Output:

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

14)	FJFo
	south Cim Kay, in fron [7] int size ) { (int size) i < size ; i-1+)
	if (frame [i] == Key) ver
Vet	unho;
3	
travel 1	- Possible 1 V 66
VOICE 1	To fife like payer [7, int h,
	int frams [7) {
with a	The Farmer of four
for C	and transfile the
	M:= 0
101 (4	) US + Kunssize; it+1) {
	nt=0; u < francis; it+1) { franc [i] = -1;}
tov (in	t i=0 / i < h; i+t) s
tov (in	t i=0 / i < h; i+t) s
tov (in	t i=0; i < h; i++) 5 (! secore (pays (i), fran,
tov (in	t i=0; i < h; i++) 5 (! severt (pays [i), fran, fransize)) 5
tov (in	t i=0; i ch; i++) 5 (! Secret (pays [i), fran, framsize)) 5
tov (in	trans [i] = -); }  trans [i] = -); }  (! Secont (pays [ii), from,  from [from ] = payed i];  from = (from +1);
tov (in	t i=0; i ch; i++) 5 (! Secarch (pays [ii), fran, framsize)) 5
tov (in	trans [i] = -); }  trans [i] = -); }  (! Secont (pays [i]), from,  from [from ] = payed i];  from = (from + 1) y. from si  foults; + 1;
tov (in	trans [i] = -); }  trans [i] = -); }  (! Secont (pays [i]), from,  from [from ] = payed i];  from = (from + 1) y, from s;  foult; + 1;
tox (in if	t i=0; i < b; i++) 5  [! secarce (pays [i), from, from [from ] = payage i]; from = (from +1) y. from si fount; ++;
tox (in if	trans [i] = -); }  trans [i] = -); }  (! Secont (pays [i]), from,  from [from ] = payed i];  from = (from + 1) y, from s;  foult; + 1;

Enter the size of page :7

Enter the page strings: 1303563

FT FO Fauts: 6 Mits: 1

15	) LRU and optimal
	int findapti hall int pages 1), is from
	3 (Six mar) the
+	for line i:0; i < framsize; 144) ?
	int i
1	for (y=index; y < h; y+1) {
	4 (fran [i] == pages [i]
	for theret=j;
+	pos > 1;
+	3
1	buch'
1	>
+	2 if (j==n) kann j;
	Vatora (po) == 0) ? 0: po 5;
+	7
	void timular dry (
	void stimular dru ( in page [3, the b,
	int for si
	int from [from or ], firm [france
1	

	Mary Section (1)
/	
//	for I
	for (int 1=0; i e ( vanesis; 1+1)
	fine[i]=0; 3
	for (ind 1=0; i(h; 1+1)?
	alled pas = 2 15
	all pos: search ( pages [i],
	Trame, francis,
	1 1 105 = = - ]) 9
	int leave=0;
	for [ind j = 1, jet tvan sie,
	j++) {
	ff (the [; ) ( tin lucan ]
_	3 barsi;
_	from Lbout ] = pages [i];
_	lime [ clear) = 2;
	(ounl ++;
	?
	else & hits H;
	2in [pos =1;30
$\overline{}$	3
_	printf("Lev faults Y.d , Lits Y.d.,
-	print + 1. Eko fames
-	count, nels);)
_	
-	

	void stimulus optimal (ibs pays, ind my
	jus franc [ francis 27, 10 ml = 0, hits=0, hit
	franc [ii]: -1;
	for (int i=0; j (h; )++);
	if (Sarah (payer i), from, francisco)
	in- judg = -1;
	too lind j-6; j L+ van sizit
	if (from [j]==-1) {
	index = y;
	bkesh;
	3
	3
	if (indox ] = -1) from [indox]=
	Jud Juday = -1;
	for ( ind j = 0) j (funcia) it)
_	indot: ii
_	b kup,
	3
	1
-	it (4mor) = -1) 5
	from CindOc3 = payerCi7.
	The House of Advices

	Proper Ed S
_	
_	
_	olic §
	in replacements: flood optimed ( payer,
_	frame, n. D framera);
	frame [xeplosemen i dx] = pop [i];
	3
	10 hut 14. 3
	elge & Lits 14; 3
	3
	prim f l" Operina fault 1.a, his 1 a"
	(out, Hill);
	( Visit )
	3
_	
	G/p+
_	Olv
	es - collect
_	and there
	Enter two siz : 12
	Enter page string : 70 2 20 3 6 422
	From no of of from 13
	Optimal Pauls: 7, Hits:s LRU Faults: 6, Hits:3
	1.00