# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



# LAB REPORT on

# **OPERATING SYSTEMS**

Submitted by

VAIBHAV KUMAR (1WA23CS027)

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 Vaibhav Kumar (1WA23CS027), 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	84-93
	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:

```
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
```

3/25			- Y	m - 1				
	20.614		DOXO	rom	dowin	wate .	m yell	Section
		Manage 1	70	and .	C.P.O. WA	Photo Male	mid later.	il.
	vim	The	d	Luxana	round	time	Cardino	WH
	0.30)							-6
_	1) FLES	1 Ero						
	2) (JF	COL	AF 60	mey	at ise	ore)		
	4100	9,51	LOYICA	166	18V1 =	)		
	Process	AT	вт	СТ	TAT	wt Ke	iponal Tin	00.8
	fr	0	7	7	7-0	р	0	
	P2				10-0		7	
	Pz		4	14	14-0	10	10	
	P4-	0	6	20	40-0	14	14	
	7 -				*			
	L Ga	nt.	chart	7				
		1 1						
	PI	72 1	P3 111	20				
	0	+ , 10	Lo	a U				
								_
	Code.							
	Hunel	ude	extedio	·h)				
	100000000000000000000000000000000000000		-					
	# I Ind	rede	coldi	ood. h	/			
			21					
	unt main []							
			,					
			t n;		11-		L 170.34	12
		unt	at	100]	bt (100.	15 (100)	, tat (1005,	will

print ("fed" dn);  Joseph ("fed" dn);  Joseph i Joseph i Salu she arrived)  for she process (fed)", i+1);  Scarf ("fod" dat lil);  Jor (unti-0; icn; i+1) {  - prints ("In Enter the bus time  for she process (god)", i+  Beamf ("fod", dbt(i));  Int sum: atloj;  Jor (unti-0; icn; i+1) {  2un+=bt[i];	Page No.:
Joseph i= 0; icn; 4+1)  print("   n Enter the arrival)  for lant i= 0; icn; i+1) {  print("   n Enter the bus teme  for the process (900)", i+  Beamf 1" fod", &bt(i);  Int sum = at[0];  for (unt i=0; icn; i+1) {  2um+=bt[i];	print ("Enter the no of processor to ent
scanf (" ) od" dat (i);  for (unti=0; ien; i++) {  printf (" In Enter the bus teme  for the process ( god ]", i+  Beamf (" dod", Abt(i);  Int sum = at[0];  for (unti=0; ien; i++) {  zum+=bf[i];	Joseph i= Oicen; 4+) {
for (unt i= 0; i <n; ("dod",="" ("in="" (god)",="" (unt="" .prints="" 2um+="bf[i];&lt;/td" 8camf="" abt(i));="" bus="" ents="" for="" i="0;" i+="" i++)="" i<n;="" jut="" process="" sum="at[0];" tems="" the="" {="" }=""><td>for the process (40 d ]", i+1).</td></n;>	for the process (40 d ]", i+1).
prints ("In Ente the bus teme  for the process [90d]", it  Beams ("40d", Abt(i));  Int sum: at[0];  for (unti=0; i(n); i++) {  2um+=bf[i];	escanf (" of od" is was
Scamf (" dod", Abt(i]);  Junt sum = at[0];  Jor (unt i=0; i2n; i++) {  zum+=bf[i];	for ( und i= 0; i < n; i++) {
Scamf (" dod", Abt(i]);  Just sum = at[0];  Jor (unt i=0; i2n; i++) {  zum+=bf[i];	for the process ( of od ] ", it
Jor (unt i=0; 12n; i++) {     zum+=bf[i];	Beamf 1" dod", AbtCiD;
for (unt i=0; 12n; i++) {     zum+=bf[i];	}
	(for (unt i=0; i2n; i++)}
ct ZiJ = sum;	ct ZiJ = sum;

	Same -	
	for ( and i= 0, i en; 1++) {	
	1	
	und til = cttil - attil;	
	\$	
	Jorlanti=0; Kn; 1++) }	
	O .	
	unt [i] = tout [i] = bf[i];	
	1	
	3	
	Voy lant i= 0; ien; i++)	
	of Inte	
	The India	
	Mark our wit	
	flour TAT, WT;	
	1/4/1	
	Yor linting; icn; i++)}	
	TAT+= tal [i];	
	WT+- wt[i],	
	3	
	1	nlin
	TAT- ( Use 12 TATI	0/12
	TAT=(float) TAT/n;	
^	WT = (gleat) WT/n;	
16	1 minklens T	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

## =>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)</pre>
    {
        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++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[i], bt[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
                7
                                       0
                                                0
       8
               3
                       14
                               6
                                       3
2
                                                3
3
               4
                        11
                               8
                                       4
                                                4
4
       5
                                15
                                               9
               6
                        20
                                       9
```

## => SJF (Preemptive):

#### Code

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

```
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
3
        2
                9
                        26
                                24
                                        15
                                                15
4
        3
                5
                        10
                                7
                                        2
                                                2
```

	Page No.
6)	SJF
	Code.
	Himolude ( stello h)
	H include coldbool h >
	where process
	5
	unt at;
	unt bt;
	int ct;
	int tat;
	unt wt;
	unt stout semi,
	\$ ps 2100];
	int main ()
	Yout used TAT = 0, votal wT=0
	bunt ni
	une completed = 0;
	bool is-disted TIOUT = & false?
	int aus t=0;
	int men = 99999999;
	unt men unden = -1
_	

printly ( " green the number of process burker: "); scanf ("dod", (n); for contieo; icn, i++) frint 1" In Eneu arrival tome process & can ( "dod", 4pstij. at ). By (anti=0; icn; i++ Scanf ( "dod", 4.ps (i] . bt). untile (completed 1= n) museumum = 215 88 9257.

Joseph is 0; len, is +)  If  if (psci) at c = our + sclice usited,  af (psci) bt c minimum)  {  minimum = psci bt;  min-Inden = 1;  if (psci) out c ps [min-indersist  minimum = psci] bt;  minimum = psci] bt;  minimum = psci] bt;  minimum = psci bt;  cust ++;	of (psci): at c= cur_tell is usited;  of (psci): bt c numerous)  framewore psci]: bt;  num-inden = i;  if (psci): bt == numerous)  cif (psci): bt == numerous)  if (psci): at c ps [min-indental  min-indental)  if (min-inden == -1)	Same -
of (pscis-at c= cur + 661 is usited)  of (pscis-at c= cur + 661 is usited)  of (pscis-bt = numerous)  of (pscis-bt= menumaro)	of (psci) at c= cur_tellients the  of (psci) bt c numerous  numerous = psci ] bt;  num-inden = 1;  of (psci) bt == menumous  of (psci) bt == menumous  of (psci) bt == menumous  if (psci) at c ps [min-indiago  min-inden = 1)  of (men-inden == -1)  cust ++;  cust ++;	Justin 1= 0; (cm; 1++)
Missis bt a retriend of the manufacture of the scil. bt:  when - I reden - I;  of (pscil. bt:= menumum)  of (pscil. bt:= m	Jeffelis bt andrewnons)  I memorial political state of the memory of the still state of the memory of the memory of the memory of the memory of the minimum of the memory of the minimum of the minimum of the minimum of the memory of the mem	N/
I ps(i) bt a retriering  I minimum = ps(i) bt;  run-Inden - 1;  I (ps(i) bt == minimum)  I (ps(i	I pscis bt a renormans)  I menimens = pscis bt;  when - Inden - I;  I (pscis at c ps I meninders)  I min-indens;  I min-indens;  I (men-indens = -1)  I ceuset ++;  I ceuset ++;	of coscil-at c= our + cel is usity
f numerous = ps(1.bt;  num-inden = 1;   if (ps(i).bt == minimum)  cif (psti).at ( ps lmin-indential  minimum = ps(i).bt;  min-inden = 1;  }  if (min-inden = = -1)  x  cust ++;	f numerous = pSCJ. bt;  Num-Inden - 1;   of (psCi]. bt == minimum)  cift pstid at c ps Lumin-industry  minimum = ps(i]. bt;  min-inden = 1;  }  f (min-inden = -1)  g cust ++;  }  cust ++;	of (ps(i) bt a ninemany
(ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (min=indin== ps(i].bt)   (min=indin==-1)   (min=indin==-1)   (cust+++,	(ps(i).bt:= minumum)   (if   psti).at ( ps [min-industal   minumum = ps(i).bt:   min-inden = 1)   (min-inden = = -1)   (cust ++;	(
(ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (ps(i].bt== minumum,)   (min=indin== ps(i].bt)   (min=indin==-1)   (min=indin==-1)   (cust+++,	(ps(i).bt:= minumum)   (if   psti).at ( ps [min-industal   minumum = ps(i).bt:   min-inden = 1)   (min-inden = = -1)   (cust ++;	mun-toden ij
	min-inden=1,  I (min-inden=1)  I (min-inden=-1)  Ceyst++;	3
	minamens = ps(i] bt:  min-inden = 1,	of (psci). bt= menuman)
min-inden=1,    (mm-inden==-1)   (cust ++;	min-inden=1,    1	
}    (men_inden = = -1)   (ceyst ++;	}    (men_inden = = -1)   (ceyst ++;	
}    (men_inden = = -1)   (ceyst ++;	}    (men_inden = = -1)   (ceyst ++;	minimum = ps(1) bt
}    (men_inden = = -1)   (ceyst ++;	}    (men_inden = = -1)   (ceyst ++;	min-jnden =1)
Cuyst ++;	Cuyst ++;	)
Cuyst ++;	Cuyst ++;	1
Cuyst ++;	Cuyst ++;	
Cuyst ++;	Cuyst ++;	7
Cuyst ++;	Cuyst ++;	1
Cuyst ++;	Cuyst ++;	1
3	3	ul (mm-inden = = -1)
3	1	Y Y
1	l whe (	cerset ++;
)	ulse (	1
	ulse (	
ulse		ulse (

ps (min index) - start to cur to & struct + pscmm.indust bt. pstmin-indent lat: pstmin-indent t gs (min-indus) at; utotal to 1 += ps [min inden ] fat; 125 (men-inden) wt=ps com-inden) for - pacmen\_inden] bt totalwit +- pstmin-indent ut; is-visited then indust = true. completed ++: printy ("In therage TAT: \$. Af ms", total TAT | n); friends ("In Average wt don't mstr" John WILh ):

#### 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):

#### Code

```
#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++) {
       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;
}
```

	Lab-Prome
Non	Freemptive Priority
Co	Preemptive Priority Xcheduling
Ħù	nclude < stdio. h>
# 3	nelude ( lynity h)
w	nut Prossis
	int id, at, bt, priority ct, tat, upt.
-5;	intid, at, bt, priority, ct, tat, ut;
· un	+ main() q
	prints (" renter the number of procures" ecans ("ded", (n):
	is truck process ps(n);
	Yorlunti=0; icn; itt)
	psc:1.id=i+2
	prints (" Enter arrival Home: ") scary (" tod tod tod" 1 (ps is at)
	aps (i). bt, 6 ps (i). prior
	Boxt By providy (ps, n);
	ent lan Hour = 0.

· (factor) = 0; icn; 1++)

```
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
                BT
                        Priority
                                        CT
                                                 TAT
                                                         WT
                                                                 RT
1
        0
                4
                        2
                                         14
                                                 14
                                                         10
                                                                 10
2
        0
                10
                                         10
                                                 10
                                                         0
                                                                 0
                        1
3
        0
                        3
                                         17
                                                 17
                                                         14
                                                                 14
4
        0
                12
                        4
                                         29
                                                 29
                                                                 17
                                                         17
```

# => Priority Scheduling (Preemptive):

#### Code

```
#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
2
Process 2: 0
Process 3: 0
Process 4: 0
Process Arrival Burst Priority
                                                  Completion
                                                                       Turnaround
                                                                                           Waiting
                                                                                           8
16
          0
                                                  16
                                                                       16
          0
                                                  18
                                                                       18
Average Waiting Time: 6.75
Average Turnaround Time: 11.25
```

wold sort By Prisvity ( struct Process (5 ( ] unt 1) struct process terms. for (un) i= 0; (cn-1, i+1) Yes land jeith; jen; jet) itemp=15[17; ps(i)= |ps(17:

Prempersive Princity scheduling Il melede coldio. h> H include (directs /1) istruct process unt id , vat , at , bit , priority , U , test in main [] Josephi=0; icn; itt ps (17. id=1) 1" Enler arrival time bel-remainer (i) = ps ii J. kt

int completed = 0, cens. Time = 0 yeo at to tal TA 1-0, total WI = 0; whell ( wrongle d < n) In min- inden = - 1; and min liverty = INT\_MAX. yor (anti=0; icn; i+t) uf ( psci) at 6= and Time be if 1 psc 13. priority confriority men\_inden=i uf (min. inden 1 2-2) be-romaining Imen-inden ] - .. curr-Time ++;

uf 161- remouning I min indende ps Emin-irdinded = cure To pstrumindent tout -John TATE psemmindent total WI+-pcZminindny. completed ++; printel (" | n Average TA T: 1), tatal TH (11) hounge ut. of ord 1 m11 (chun ()

#### => 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
                                       25
                       15
2
        5
                        15
                                       20
3
        7
                        16
                                       23
        3
                       12
                                       15
Average Waiting Time: 14.50
Average Turnaround Time: 20.75
```

Kound Kobin # unclude coldio h> struct Mouse in id, at, bt, it, cl, start, unt, In mount () struct processpring. 15 Cil·id = 1++; frint ( " Enty other are in al tens 4RT 11); scand 1" ded ded ", 6 pstil ad, 6 pstil -t. with completed = 0; cent-Time 0; follow bold TAT= 0, Untry WT= 0

Sur Sur	\$
det interior (Child)	grind " Rusing TAT; \$2.3 1", Inda Tr Tfafe  which ("Rusings with forf", Solar with
Jone C.	}
4 (file) it > quentum)	
petitet a guarriera	
ulyk	
printed a print the	
osminii y li nu ushdur v y li nu ushdur v y li nu	

#### 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%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;
```

Output:

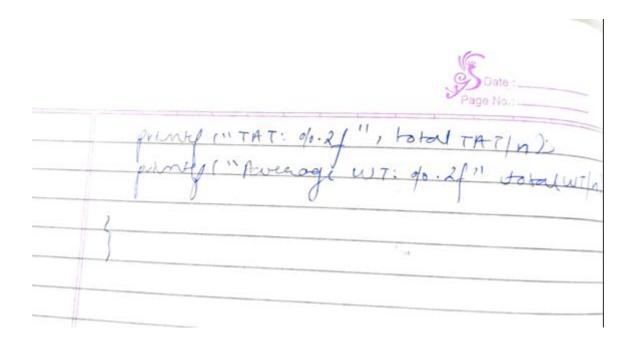
```
Enter number of processes: 4
Enter Burst Time, Arrival Time and Queue of P1: 2
1
Enter Burst Time, Arrival Time and Queue of P2: 1
2
Enter Burst Time, Arrival Time and Queue of P3: 5
1
Enter Burst Time, Arrival Time and Queue of P4: 3
2
Queue 1 is System Process
Queue 2 is User Process
Process Waiting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

36

3/04/15 Multilevel Queue Educations H unclude caldo h) Street Process & un al. bl. d. vat. vid. out, gra en mount) unt n, quantum 1; sound (" fod", Enter no. of processes "1). where process ps (n); int be remaining Co 7; Yorlantico; icnil++) 152:1. id = 1+1; of " Enter AT, BT, Graces 1/60 :" bet- (communent [i]= [s[i], b)

Enter quartum for Round Rober 1. une conservations = 0. completed = 0. Glose Total TATED. Using UT = 0 unshale ( completed (n) unt don'=1; for lant 1: Discount 1) scanaining 21) > 0) done = 0; U (bt-reomaining Ei)>quarter cuy Timm += 4 pranterm! be reamoung 21. = quarrant Cur7+= Guantums! 1. reamouning Tiled: \$5 Tilet = affere Time;

If (done) break; while (completed on) unt dans; forcumies; ien itel done=0; clue time += bt\_remaining zi]: bt-remaining (i): 0; ps lizet : copy Time; ps (i) . it at = ps ? i) . d -pcli) at ps li) w+ = ps 212. 70/pscij.by; of (done) & break; }



#### 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;
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;
}
```

## Output:

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

RATE MONOTONIC une pldg period, bens, comas task; ent ged (ant a, ent b) seewn 6: =0) a ; ged Cb, ago by; un lem conta, int b) xchun axbl ged (a,b); ent find HP (tash tasks [], int n) unt chyper = tasks 20] period: yor (unrist; icn; i+1) dryper : dom ( dryper, tashti? period) return eypos:

100	
	Some Contraction of the Contract
,	Pageto
ار	word sorthyperiod 1 tack tashs [] in for ( und is 0 , it no is 1)
	O The last of
_	for land is on it ni i+1)
	V
	for land jult 1 jenijet)
_	3 1 1-11/12
	Tosk to p into
	1 100 100 100 100 100 100 100 100 100 1
	Tosk by probat
	Jacks (ill In )
	Jack 1117.
	Tosk temp : Jasks [i]; Vashs [i]: Jashs 42; Jashs []: Jermp,
	,
	thank, TN
	Vacks [i] remaining = Jacks [i]: buch
	1
	1
	effect calculate (PU utilization (Part tank)
	in ny
	{
	efleat ( and j=0; icnii+1)
	Ψ.
	unilesation = (closed) Jackeli)
	bust sates (il period)
	Yeturn willigah? or
	2
	<u>"</u>

	flow well-africe = calcular
	float white hold = n+ (
	) willen some of
	Patient = Cal
	Lacerday China
_	float ( you way and
	There had
	Par il
	float three hold = no ( power Liveter) - 1 35
	Millisonia,
	telf In "
	less in the second
	the selection of the se
-	front 1" ( out weitigation telf In").  friend 1" Schedulability &
	1º Af lall
	friend 1" Schedulability to 14/11", of (utilization (= threehold)  wreturn 1.
	Carrier Continue
	1) three
	Unal.
	releur 1.
	61
	else [
	(chun 0;
	)
	word rate Monotonic ( Task tashs ?? , unt n,
	confignic Clash tashill, and n.
	unt sime )
	200M
-	
	print (n) by
	prend (" In Late-Monatone schoduleny)
	U.
	Mark 1117 - 15 - 15 11
	print ("Time (n talk In").
	COLLABOL (CAMPSINI)

used scheduled = - 1; for completon ich; (+1) of wome of walk (i) period == 0) of Justis umarrieng: Morlewi=0; icn; itt) of walks (17 remaining > 0) scheduled = i. a) (scheduled 1 = -1) salks [ selve dulad] . remounty also & print (" to 1+ 2d ( n 11, veens)

#### => 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++)</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; <math>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; i++) {
            if (current time < tasks[i].deadline && tasks[i].executed
< 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++) {</pre>
                printf("\t");
            }
        printf("\n");
    }
}
int main() {
    Task tasks[] = {
        {1, 20, 3, 20, 0},
        {2, 5, 2, 5, 0},
        {3, 10, 2, 10, 0}
//task
    };
    int num_tasks = sizeof(tasks) / sizeof(tasks[0]);
    int total_time = 20;
    earliest deadline first scheduling(tasks, num_tasks, total time);
    return 0;
}
Output:
```

```
Task 1 Task 2 Task 3
Time
0
         Exec
1
         Exec
2
                 Exec
3
4
5
6
7
                 Exec
                          Exec
                 Exec
         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);
    } 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
```

1	Producer (	Constru	Pers utality
H J	nelude coldio.h		
in S	mechan: )	Yell = 0	ecapte 3, 200.
- 3	v signalia	nts)	
Viold	return (+1		
	mutero:	mant (m	ul:
	prints 1"	Produced is	cempy ); com: god", n);
word	l consumer()		
1	mulen = war		

compley = signal comply to muter signal ( muters), () main () don't director while (2) print ("inter upour choice:"). childre (chorce) case 1: st ((muter == Z) EC (cempty )= 0)) break; 50

	- S	1
	Case 2	
	consumer ()	
	elle	
	prints ( auffer surpty ").	
	break;	
-	cases:	
_	exit (0).	
	default:	
	Q .	
	printf (" Emalid choice"),	
-	Private Company action	
	}-	
	1	
-	5	
	yetren 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 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
```

vlant -	S.
10/00/25	aring Willeghin
	# unclease a pthread. h>
	# smelerde & samophore h?
	thended Lateld h >
	4 define N 5
-	# define THOUSENG &
	H define MUNGRY
	Exime EATING O
	+ define LEFT Lphnam +4) to N
	# define LEFT Lyhnam +4) to Ny # define KICHT ( principal + ) to Ny
	w. L. state TAIT.
	un you EN] = { 0,1,2,3,4};
	sem- mutin;
	sem-tscNI;
	word test ( out ph num)
	if (what a Commum) == HUNGENECE Start CLECTI: FRANK
	State CLECTI: FATTING
	CL STORE TRICIANT ] 1 = EATING
	1
	istate [phnum]= f. p. min.
	esterp(x):
	printy 1" philosophy dod takes
	phramma + 1, 1, FF + +1 phrames

	print i " Pricto copinis tod in Eastury i provinces);	
	sen photo my	
	The Colonia to Eating	
	Then II. I (Material);	
	word value	
	tork (any	
	Bearn-wal 14 mutes	
1	Catalog I muster	
Lunx	print 1 Bulosephon led in any	ww
		1)
	yella, parasana (1)	
	- on the 11	sch
	= U CUD ( / + f )	e 1 +
	sleep (2);	الما
	7	
	)	1
	unid	
166	word putfork ( in phroum)	1
TING)	sem wast ( & musen?;	
	sem walt ( f mulen?;	
	printy ( Phibsopher ded gutting fork do de	]
	do d damp! ), program +1,	
	ford down "), phonon+1,	
	The state of the s	-
	V.	
15	prince commended as shenring	
	(h", phrium +1)	
milej		
	Usest (LEFT);	
	rest (RICIHII)	
	rest (RICIFII)	

word \* perilosopher ( word \* num)

word \* perilosopher ( word \* num)

while (1)

while (1)

where (1)

rate - fork(\*1);

where fork(\*1);

put fork(\*1);

#### 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:
```

63

```
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
4 3 3
Enter Available matrix:
System is in a Safe State.
Safe Sequence: P1
                            P4
                                    PØ
                                           P2
```

-37
12/4 (11/4)
Sab 5 ( carkers prosum)
Hundrede criwo is
And meaning!
milet 1" Toda du me d'orocurer
print (" Toda do no of procure and
8 cary 1 10 A 10 A, 601 0 10 1)
and allo sin I (m I, manifa I (m I, and)
ant finish [n], work (m), coffeepens
fring ( " ench Allocation Hotein In !
for farriend 1=0, 10 min 1+0)
Redard (1/4) di la Machocci [6]
young (" gute where mouthing (n))
fortent(=0, jemijet)
grand (" and mailable known "
401/(un =0, cmijet)
"scenft" ded", 6 moul ( )
for lastic Oilemilet)
work Lil 2 avail (i)
Yorrestien, icm, itt)
finishEJ=0;

005 changed = 0. for white busers 199) sent dentilicaries 1 allocailli. france Sunkall amplicable O. break. of Loanthocated yorlantheo, kemiket) U weekelk] += allectillhe] Reference Count ++2=1; while (changed)

```
Just deadlock - 0.

Jest 12 femile (1)

Oleadlock - I.

proff (" Femile (1))

Left deadlock (""")

(print) ("Eptem as arreafo ross ("

print) ("

print) ("Eptem as arreafo ross ("

print) ("

print)
```

```
Cupid

Suter successes of fusions and resource 53

2 1 2

1 0 0

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 1

2 0 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");
    } else {
        printf("System is not in a deadlock state.\n");
    return 0;
Output:
Process 1 can finish.
```

```
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 45td io. h>
    # Include ( stdhoolin)
    # detine Ps
    #diffine R3
    in main Of . . .
         int finish Ep3=404; .
          int WOAR [R];
          int need crater = 5
          € 7, 6, 33;
           13,2,23,
          4 9,0,27,
           12,2,27,
           1 4,3,32 35
         Intallocation EPSIRI . 2
           £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 LR; i++) f
              work cis = quoilable cis; 3
```

```
hool deadlock fala;
     Int count = 0;
     While (count 2P) &
        hool found = fulk;
          tor (Int P=0; p < 1; p+){
           16 (Flaish 103 =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) {
              workes += allocation costs; 5
           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

#### Code

```
#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...
```

J	1 0	,	
0	ab frogram.	. 6	
	V		
	Best fell		
Hunslud	estation h	>	
whet 61	lock		
(			
tind	610de 2001		
	block size:		
	is-fre:		
2.	V		
1.			
wheel this	ul + filano: + file_siz		
Su	& Silano		
مار	+ Rele_ 813		
3;	0		
1			
word but	Est-Costnet !	Slock blocks (	1 1000 11-
where	ed till file	scl, unt no fel	LS ?
pinte	( of la memory	management Lit (n 1);	Buten
1 0	Buty	WINDS	14 81.
printel	CELD. MI	111t Ellevis	new In
1	Block si	ze 14 Frager	
Jak	1:00 ic	n-files; it t	7
90714	Well- at It		
7	but helt 1	it block =	-1:
	but makes	fragement	: 10000
-1	of Lundie	fen-block	us it t
1	5		V

printy ( " fed 41t fed 1+lt dod 1+l+ 1 in " ga blocks Chest-got block I. block. min. fragment) util

Worstful and Firstfit # unclude coldio.h. Struct Block Unt block-no: unt block size; und lafree? whent till in file no; word firstfit i street Block blocks Ch works block for ( ent is 0; ic on-files; iff) int allocated = 0;

( Seri — )
dos (invij=0) je m-blocke, je+)
of Chloches [ ] . refree & blocket [ ] block-size
files List yell - oze;
blocks() I is ofter = 0;
printy ( feles Es. file - no file : Cis file
blocks(13. block no blocks(1) block of fragment);
allocated:1;
breat;
1
uf (! allocated)
privat (" dod it abd it (t , files Cil-file me giles Cil, (file - size);
5

1000000	word months ( womet black blocks)
-	in me files []
-	July negative
-	1
-	I william management
	front in themory management water
	picted of
	fre land (= 0) ex no files (184)
	Y .
-	unst moretafiet block : - 1,
	man-fragment = 1;
	fullwood = 0: je n. Starts: j++)
	of telocks of the first bl blocks (1)
	of Chlore GI on for the blocking)
	7
	and decree
	the Majorat - Clarks GI block so
	int fragment = blocks of I block sig - filestil file size,
	Il Garment
	of (fragment > man - fragment)
	man to
	man fragmen : fragment.
	- Proposed
	,
	1

	( bry _ )
	of (worst-fit-block). 2)
	Elpotes worst fet block ] 12-free = 0.
	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
	ulse
	privil (ifilested yell-no, filestides
	1
	1
	word went Block ( street Block blocks []
	int n-blocked)
	<b>€</b> ′
	doe cant i=0: ic n_blocks, i++) }
	for cant i=0: ic n-blodes; i++) {  blocks [i]. is-free=1;
	3
-	
	}

## 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

"LKU and cotingal # unclude coldiatis I unded eddle is return - 1; break)

vetur (pos = 2) word simulate the land pages 13, and my ent framelikames ize I, deme (framesjæl), You lune i- a, inframeway : it t

frame Edward J pages Cil count ++; elses July (1) Aime Gost=i; printy 1" Lev page faults: of od, wold circulate optimus (int pages, into int frame (frame orze I count =0, hours You (int i=6; i effame 8 13e; i++. for contien; it if (search (pages Cil., from, in inden = -1;

7	Yoursellie-11
E	if ifamily : 121
	break;
+	abel
I	Jost replacement fine
	dearne Cuplaniment indenti
	Count ++
1	
	else (
1	
7	