

Department of Computer Science and Engineering
Final Examination Fall 2023
CSE 321: Operating Systems



Duration: 1 Hour 45 Minutes

Total Marks: 40

Answer the following questions.
Figures in the right margin indicate marks.

- 1. CO4** a) A super shop has launched year-end sales on all their products. To avail the offer so many customers went there and purchased products as per their preferences. The issue occurred when they started the procedure of bill payment. There are 3 counters for paying bills but the number of customers waiting for completing payment is 50. [3]
- Explain** with proper logic, what issue has been raised in the above scenario and what will be the approach to provide proper synchronization to the issue according to the problem statement.
- b) For the upcoming PMCO finals team “xyz” has arranged training sessions for players. In a training session a player needs to use a set of headphones and a mobile device together. The team can only facilitate a set of headphones and a mobile device to players for training. In a particular session 1 hour left for 2 players Alex and Zyll. Each will get a 30 mins slot. But somehow Alex has captured the device and Zyll has captured headphones at the same time and that is why nobody is able to make any progress in the training session. [2]
- Logically explain** what issue has occurred in the above scenario deadlock=
- c) In a system, following conditions are present.
- There are 3 processes: P1, P2 and P3.
 - There is a semaphore, $s=2$.
 - Ready queue is in the following order, [P1, P2, P3].
 - CPU allocation is managed by round robin scheduling algorithm with the time quantum of 6 ms.
 - Each statement takes 2 ms to execute.
 - Critical section contains 3 statements.
 - Remainder section contains 2 statements.

The structure of process P_i in solution using Semaphore:

<pre>wait(s){ while(s<=0) ;//busy wait s--; } signal(s){ s++; }</pre>	<pre>do{ wait(s); //critical section signal(s); //remainder section }while(true);</pre>
---	---

Complete the table given below for processes P1, P2 and P3 using semaphore.

[5]

Process 1	Process 2	Process 3



2. a) We have various ways to overcome deadlock in a system. Among these approaches is the strategy of ignoring it and relying on system restarts to resolve the deadlock. Despite the need for restarting the system, this method remains popular. **Discuss** why this strategy is commonly employed and **mention** the type of system that may utilize this method

[3]

CO4

- b) Suppose, in a workplace, we have a set of resource types, $R = \{R1, R2, R3, R4\}$ and a set of processes, $P = \{P1, P2, P3, P4\}$. **R1, R2, R3, and R4** have **2, 2, 2, and 2** instances respectively.

- P1 is holding 1 instance of R4
- P2 is holding 1 instance of R1
- P3 is holding 1 instance of R1
- P4 is holding 1 instance of R4
- P4 holding 1 instance of R2
- P2 requests 1 instance of R3
- P2 is holding 1 instance of R2
- P1 is requests 1 instance of R1
- P3 is holding 1 instance of R3
- P4 is holding 1 instance of R3
- P3 requests 1 instance of R4
- P4 requests 2 instances of R1

Construct a resource allocation graph for the above scenario and **identify the cycle (if any) and decide** whether there is a deadlock or not.

[4]

3. a) Arrays are stored in contiguous memory locations to optimize access to array elements, yet allocating processes in contiguous memory locations is discouraged. **Explain** why this is not recommended in terms of space complexity.

[3]

CO5

b) A system with an associative lookup time of 7ns, and memory access time of 59ns, what should be the approximate hit ratio to achieve Effective Access Time of 92ns? [3]

c) Assume that, page size of a process is **8 bytes** and size of the main memory is **72 bytes**. Logical memory and page table of the process are given below.

Logical Memory		PMT		Main memory
Page #	Data	Page #	Frame #	
P0	ab	P0	2	
P1	bc	P1	6	
P2	cd	P2	7	
P3	de	P3	13	
P4	ef	P4	11	
P5	fi	P5	5	

- How** can the user's view of memory be mapped into the main memory? [1]
- Find out** corresponding physical addresses of the following logical addresses – **18(10010), 44(101100) and 27(11011)** [3]

d) If the page size is **9 KB**, **how many** frames will be needed in Main memory for a process size of **83,645 Bytes**? Is there any **internal fragmentation**? - If yes, **calculate** the value. [1 KB = 1024 Bytes] [2]

e) In a particular time, the snapshot of Main memory given below for dynamic partitioning where gray portions of the memory are representing occupied spaces. Apply worst fit and first fit algorithms to place processes with the space requirement of **P1=600k, P2=400k, P3=348k, P4=200k, P5=52k, P6= 100k and P7=72k** (in order). Explain which algorithm makes the most effective use of memory? [5]

800K	600K	120K	100K	400K	522K
------	------	------	------	------	------

- 4. a)** Consider a computer with a main memory that has 3 frames and page reference string of 0-7 page **[0, 1, 6, 6, 4, 0, 0, 5, 5, 4]**. The page reference string represents the order in which the pages are accessed by a program. **Apply LRU & OPT** algorithm to **simulate** the page replacement that occurs when the main memory can hold at most 3 pages at a time. **Record** the number of **page faults** and compare the result. **Mention** which algorithm performs better in this scenario. [6]

CO5

Department of Computer Science and Engineering
Final Examination Fall 2023
CSE 321: Operating Systems

B

Duration: 1 Hour 45 Minutes

Total Marks: 40

Answer the following questions.
Figures in the right margin indicate marks.

1. **CO4** a) In an office there are 10 employees. All the computers of the offices are connected to the internet by wired connectivity. But due to the limitation of bandwidth, office authorities allow only 5 devices to be connected with the wifi at the same time. On a particular day, 2 devices are already connected to the wifi and 6 more employees are trying to connect their devices to the wifi at the same time. **Explain** with proper logic, what issue has been raised in the above scenario and what will be the approach to provide proper synchronization to the issue according to the problem statement. [3]

b) For the upcoming PMCO finals team "xyz" has arranged training sessions for players. In a training session a player needs to use a set of headphones and a mobile device together. The team can only facilitate a set of headphones and a mobile device to players for training. In a particular session a 30 mins slot has been allotted for each player. 4 players of the team have arrived for the session and a queue has been fixed based on the ascending order of their arrival times. According to the criteria mentioned above the order of the players in the queue is Action, Top, Icy and Nirzed. But they were called for the training according to the following order: Nirzed, Icy, Top and Action. Therefore, after waiting for a long period Action left the training arena out of annoyance. **Logically explain** what issue has occurred in the above scenario. [2]

c) In a system, following conditions are present.

- There are 3 processes: P1, P2 and P3.
- There is a semaphore, $s=2$.
- Ready queue is in the following order, [P2, P3, P1].
- CPU allocation is managed by round robin scheduling algorithm with the time quantum of 9 ms.
- Each statement takes 3 ms to execute.
- Critical section contains 2 statements.
- Remainder section contains 3 statements.

The structure of process P_i in solution using Semaphore:

<pre>wait(s){ while(s<=0) ;//busy wait s--; } signal(s){ s++; }</pre>	<pre>do{ wait(s); //critical section signal(s); //remainder section }while(true);</pre>
---	---

Complete the table given below for processes P1, P2 and P3 using semaphore.

[5]

Process 1	Process 2	Process 3



--	--	--

2. a) We have various ways to overcome deadlock in a system. Among these approaches is the strategy of ignoring it and relying on system restarts to resolve the deadlock. Despite the need for restarting the system, this method remains popular. **Discuss** why this strategy is commonly employed and **mention** the type of system that may utilize this method

[3]

CO4

b) Suppose, in a workplace, we have a set of resource types, $R = \{R1, R2, R3, R4\}$ and a set of processes, $P = \{P1, P2, P3, P4\}$. **R1, R2, R3, and R4** have **2, 3, 2, and 3** instances respectively.

- P1 is holding 1 instance of R2
- P2 is holding 2 instances of R2
- P3 is holding 1 instance of R1
- P3 requests 1 instance of R2
- P4 holding 2 instances of R4
- P2 holding 1 instance of R1
- P1 requests 1 instance of R1
- P4 requests 1 instance of R3
- P3 holding 1 instance of R3
- P1 holding 1 instance of R3
- P3 holding 1 instance of R4

Construct a resource allocation graph for the above scenario and **identify the cycle (if any) and decide** whether there is a deadlock or not.

[4]

3. **a)** Arrays are stored in contiguous memory locations to optimize access to array elements, yet allocating processes in contiguous memory locations is discouraged. **Explain** why this is not recommended in terms of space complexity. [3]

CO5

b) A system with an associative lookup time of 2ns, and memory access time of 72ns, what should be the approximate hit ratio to achieve Effective Access Time of 95ns? [3]

c) Assume that, page size of a process is **8 bytes** and size of the main memory is **72 bytes**. Logical memory and page table of the process are given below.

Logical Memory		PMT		Main memory
Page #	Data	Page #	Frame #	
P0	ij	P0	5	
P1	jk	P1	16	
P2	kl	P2	7	
P3	lm	P3	3	
P4	mn	P4	6	
P5	no	P5	12	

i. **How** can the user's view of memory be mapped into the main memory? [1]

ii. **Find out** corresponding physical addresses of the following logical addresses – **25(11001), 37(100101) and 23(10111)** [3]

d) If the page size is **10 KB**, how many frames will be needed in Main memory for a process size of **31,110 Bytes**? Is there any internal fragmentation? - If yes, **calculate** the value. [1 KB = 1024 Bytes] [2]

e) In a particular time, the snapshot of Main memory given below for dynamic partitioning where gray portions of the memory are representing occupied spaces. Apply worst fit and first fit algorithms to place processes with the space requirement of **P1=600k, P2=400k, P3=298k, P4=292k, P5=200k, P6= 100k, P7=44k and P8=58k** (in order). Explain which algorithm makes the most effective use of memory? [5]

800K	600K	320K	100K	400K	522K
------	------	------	------	------	------

4. a) Consider a computer with a main memory that has 3 frames and page reference string of 0-7 page [5, 5, 3, 1, 7, 3, 3, 5, 2, 0]. The page reference string represents the order in which the pages are accessed by a program. **Apply LRU & OPT** algorithm to **simulate** the page replacement that occurs when the main memory can hold at most 3 pages at a time. **Record** the number of **page faults** and compare the result. **Mention** which algorithm performs better in this scenario.

[6]

Department of Computer Science and Engineering
Final Examination Fall 2023
CSE 321: Operating Systems

C

Duration: 1 Hour 45 Minutes

Total Marks: 40

Answer the following questions.
Figures in the right margin indicate marks.

1. **CO4** a) In the research lab of a university there is a high-performing computer which can be used for research works on parallel computing. In a particular semester, four research groups are working on separate projects on parallel computing. One day four groups came together at the lab and were willing to use the high-performing computer at the same time. [3]
- Explain** with proper logic, what issue has been raised in the above scenario and what will be the approach to provide proper synchronization to the issue according to the problem statement.
- b) In a certain match of PMCO two players from team “xyz” Action and Top started a debate over the sniping role of the team. Both of them are good in long range and in the match, Top found out a sniper weapon but he has no scopes. As a result, he is unable to use the sniper for the long range. On the contrary, Action has an 8x scope but he does not have any sniper weapon. Which means he is unable to use the scope. Both of them are willing to play as a sniper and for that Top is demanding the scope from Action and Action is demanding the sniper weapon from Top. But nobody is willing to make the compromise. Therefore, neither of them can play as a sniper. [2]
- Logically explain** what issue has occurred in the above scenario.
- c) In a system, following conditions are present.
- There are 3 processes: P1, P2 and P3.
 - There is a mutex lock, available=true.
 - Ready queue is in the following order, [P3, P1, P2].
 - CPU allocation is managed by round robin scheduling algorithm with the time quantum of 12 ms.
 - Each statement takes 4 ms to execute.
 - Critical section contains 3 statements.
 - Remainder section contains 2 statements.

The structure of process Pi in solution using mutex lock:

<pre>acquire() { while(!available) ; //busy wait available=false; } release() { available=true; }</pre>	<pre>do{ acquire(); //critical section release(); //remainder section }while(true);</pre>
--	---

Complete the table given below for processes P1, P2 and P3 using mutex lock.

[5]

Process 1	Process 2	Process 3



--	--	--

2. a) We have various ways to overcome deadlock in a system. Among these approaches is the strategy of ignoring it and relying on system restarts to resolve the deadlock. Despite the need for restarting the system, this method remains popular. **Discuss** why this strategy is commonly employed and **mention** the type of system that may utilize this method

[3]

CO4

b) Suppose, in a workplace, we have a set of resource types, $R = \{R1, R2, R3, R4\}$ and a set of processes, $P = \{P1, P2, P3, P4, P5\}$. **R1, R2, R3, and R4** have **3, 2, 4, and 2** instances respectively.

- P1 is holding 2 instances of R1
- P2 is holding 1 instance of R3
- P3 is holding 1 instance of R4
- P5 requests 2 instances of R3
- P4 is holding 1 instance of R4
- P3 requests 1 instance of R2
- P2 requests 1 instance of R1
- P2 is holding 1 instance of R2
- P1 is requesting 1 instance of R4
- P3 is holding 1 instance of R3
- P4 is holding 1 instance of R3
- P5 holding 1 instance of R2

Construct a resource allocation graph for the above scenario and **identify the cycle (if any) and decide** whether there is a deadlock or not.

[4]

3. a) Arrays are stored in contiguous memory locations to optimize access to array elements, yet allocating processes in contiguous memory locations is discouraged. **Explain** why this is not recommended in terms of space complexity.

[3]

CO5

b) A system with an associative lookup time of 5ns, and memory access time of 85ns, what should be the approximate hit ratio to achieve Effective Access Time of 146ns? [3]

c) Assume that, page size of a process is **8 bytes** and size of the main memory is **72 bytes**. Logical memory and page table of the process are given below.

Logical Memory		PMT		Main memory
Page #	Data	Page #	Frame #	
P0	op	P0	10	
P1	pq	P1	2	
P2	qr	P2	4	
P3	rs	P3	11	
P4	st	P4	8	
P5	tu	P5	3	

i. **How** can the user's view of memory be mapped into the main memory? [1]

ii. **Find out** corresponding physical addresses of the following logical addresses – **11(1011), 4(100) and 21(10101)** [3]

d) If the page size is **7 KB**, how many frames will be needed in Main memory for a process size of **93,600 Bytes**? Is there any internal fragmentation? - If yes, **calculate** the value. [1 KB = 1024 Bytes] [2]

e) In a particular time, the snapshot of Main memory given below for dynamic partitioning where gray portions of the memory are representing occupied spaces. Apply worst fit and first fit algorithms to place processes with the space requirement of **P1=600k, P2=400k, P3=298k, P4=292k, P5=200k, P6=100k, P7=44k and P8=58k** (in order). Explain which algorithm makes the most effective use of memory? [5]

800K	600K	320K	100K	400K	522K
------	------	------	------	------	------

4. **a)** Consider a computer with a main memory that has 3 frames and page reference string of 0-7 page **[3, 5, 4, 6, 7, 4, 2, 6, 7, 6]**. The page reference string represents the order in which the pages are accessed by a program. **Apply LRU & OPT** algorithm to **simulate** the page replacement that occurs when the main memory can hold at most 3 pages at a time. **Record** the number of **page faults** and compare the result. **Mention** which algorithm performs better in this scenario. [6]

CO5

5. [CO1] 1.a) Answer the following question: **[3 Marks]**

3 points

In a busy computer lab during finals week at a university, many students need access to a limited number of computers for their assignments and exams. To ensure fair and efficient resource allocation, which concept of operating system can be employed to address the issue? Explain how this method can solve the problem and mention the benefits it offers.

semaphore and cpu scheduling

6. [CO1] 1.b) Answer the following question: **[2 Marks]**

2 points

What specific problem(s) of the monolithic structure were addressed through the adoption of a layered structure, and how were they resolved?

Debugging, more organized, easier to upgrade

7. [C01] 1.c) Find the output of the following code snippet. You need to type the answer in this form (as a response to this question) and show your working / tracing on paper. **Your output should exactly match with the original output. [3 Marks]** 3 points

```
int main(){|
    int a=9;
    int b=3;
    i=fork();
    if(i<0){
        printf("fork failed\n");
    }
    else if(i==0){
        j=fork();
        if(j<0){
            printf("fork failed\n");
        }
        else if(j==0){
            a=a*b;
            b=a/b;
        }
        else{
            wait();
            a=a+b;
            b=b-a;
        }
    }
    else{
        wait();
        a=a-b;
        b=b+a;
    }
    printf("value of a: %d\n",a);
    printf("value of b: %d\n",b);

    return 0;
}
```

8. [C01] 1.d) Answer the following question: **[3 Marks]**

3 points

In a Google Classroom, there are two types of users: teachers and students. Teachers create assignments with instructions and attached problem files, resulting in assignment slots in the classroom. Each slot contains instructions, the attached file, and individual placeholders for students to submit their assignments. Students can access instructions and problems from the attached files within these slots. When students submit assignments, they use designated placeholders within the assignment slots. Teachers review student assignments by accessing the files from these placeholders.

Logically explain what type of communication method was used in the above given scenario.

Message passing- as cant be shared data as student cant change what teachers write
acts like a connecting device betwn two entities

9. [C02] 2.a) Type the average waiting time and turnaround time in this form (as a response to this question) and show your calculation on paper : **[5 Marks]** 5 points

Draw a Gantt chart and illustrate the execution of the process using the **Round Robin** scheduling algorithm (**time quantum = 5 units**). **Calculate** the **average waiting** and **turnaround time**.

Process ID	Burst Time	Arrival Time	Priority
P1	5	2	1
P2	6	6	5
P3	13	6	4
P4	15	10	222222
P5	9	12	3

10. [C02] 2.b) Answer the following question: **[2 Marks]** 2 points

Due to a calculation error, P4 has received an abnormally high priority value compared to other processes. Describe the problem this situation might cause by providing a scenario illustrating the issue, and suggest how to address it.

starvation, aging with the min val deduction each time

11. [CO3] 3.a) Type the answer in this form (as a response to this question) and show your calculation on paper : **[3 Marks]** 3 points

A system has processes to execute of which are 86% parallel. The number of cores currently available is 2. Calculate the number of cores required in order to increase the speedup approximately 2 times.

12. [CO3] 3.b) Answer the following question: **[2 Marks]** 2 points

Imagine you have a text editor that is running on multiple threads and has a python code execution feature. To achieve the code execution, the editor creates a child process and loads the python interpreter as a separate program. In this scenario, should the child process be a single-threaded or multi-threaded program? State your reasons.

multi threaded child to utilize cpu as text editor isnt doing anything else

13. [CO3] 3.c) Answer the following question: **[2 Marks]** 2 points

Suppose an organization has a million employees. They preserve both HR management data including their demographic data. At an annual meeting on 31st December the CEO of the company wanted to pay a 20% bonus to employees whose age is more than 50 and achieved 90% KPI on 1st January. As the time is limited the data analyst used many nodes to make the search and generate the result. **Identify** which parallelism technique can be applicable here?

data parallelism- seperate data to act operate faster

5. [C01] 1.a) Answer the following question: **[3 Marks]**

3 points

A call center for a telecommunications company handles a constant flow of customer calls and inquiries. Due to a limited number of service providers, most of the calls are dropped unanswered. To solve this problem they implemented a waiting lobby when all the lines are busy. Customers can wait till someone from the company picks up their call. Which concept of the Operating system has been employed in this scenario? Explain how it works in the context of the operating system.

Cpu scheduling- ready queue- short term scheduler

6. [C01] 1.b) Answer the following question: **[2 Marks]**

2 points

What specific problem(s) of the layered structure were addressed through the adoption of a microkernel structure, and how were they resolved?

7. [C01] 1.c) Find the output of the following code snippet. You need to type the answer in this form (as a response to this question) and show your working / tracing on paper. **Your output should exactly match with the original output. [3 Marks]** 3 points

```
int main(){
    int x=15;
    int y=5;
    i=fork();
    if(i<0){
        printf("fork failed\n");
    }
    else if(i==0){
        x=y-x;
        y=x+y;
    }
    else{
        wait();
        j=fork();
        if(j<0){
            printf("fork failed\n");
        }
        else if(j==0){
            x=x/y;
            y=x*y;
        }
        else{
            wait();
            x=x-y;
            y=x+y;
        }
    }
    printf("value of x: %d\n",x);
    printf("value of y: %d\n",y);

    return 0;
}
```

8. [C01] 1.d) Answer the following question: **[3 Marks]**

3 points

In a Google Classroom, there are two types of users: teachers and students. Teachers create assignments with instructions and attached problem files, resulting in assignment slots in the classroom. Each slot contains instructions, the attached file, and individual placeholders for students to submit their assignments. Students can read instructions and problems from the attached file from that particular assignment slot. When a teacher creates an assignment, notification gets distributed to every teacher and student of the classroom. If a student submits an assignment every teacher of the classroom gets acknowledgement by a notification.

Logically explain what type of communication method was used in the above given scenario.

9. [CO2] 2.a) Type the average waiting time and turnaround time in this form (as a response to this question) and show your calculation on paper : **[5 Marks]** 5 points

Draw a Gantt chart and illustrate the execution of the process using the **Round Robin** scheduling algorithm (**time quantum = 6 units**). **Calculate** the **average waiting** and **turnaround time**.

Process ID	Burst Time	Arrival Time	Priority
P1	9	5	1
P2	13	6	5
P3	7	11	4
P4	11	9	222222
P5	11	17	3

10. [CO2] 2.b) Answer the following question: **[2 Marks]** 2 points

Due to a calculation error, P4 has received an abnormally high priority value compared to other processes. Describe the problem this situation might cause by providing a scenario illustrating the issue, and suggest how to address it.

11. [CO3] 3.a) Type the answer in this form (as a response to this question) and show your calculation on paper : **[3 Marks]** 3 points

A system has processes to execute of which are 82% parallel. The number of cores currently available is 3. Calculate the number of cores required in order to increase the speedup approximately 1.5 times.

12. [CO3] 3.b) Answer the following question: **[2 Marks]** 2 points

Imagine you have a text editor that is running on multiple threads and has a python code execution feature. To achieve the code execution, the editor creates a child process and loads the python interpreter as a separate program. In this scenario, should the child process be a single-threaded or multi-threaded program? State your reasons.

13. [CO3] 3.c) Answer the following question: **[2 Marks]** 2 points

In a University, students, department, admin everyone share the same data structure. Students use it for registering their course, the admin office approves the course and the account department confirms their registration, and the department assigns a faculty for coordinating everything about the student. **Identify** which parallelism technique can be applicable here?

task parallelism- different task on same student data

5. [C01] 1.a) Answer the following question: **[3 Marks]**

3 points

John, a college student, received a designing task on a project management software. To complete it, he used the windows operating system to open adobe illustrator. While he was reading the requirements from a microsoft doc, he received a prompt from an antivirus software. - Identify different types of softwares used by the user in this scenario and mention a few differences between them.

system software- op system - used by karnel, handels hardware,

application software- user interface

6. [C01] 1.b) Answer the following question: **[2 Marks]**

2 points

What specific problem(s) of the monolithic structure were addressed through the adoption of a microkernel structure, and how were they resolved?

7. [C01] 1.c) Find the output of the following code snippet. You need to type the answer in this form (as a response to this question) and show your working / tracing on paper. **Your output should exactly match with the original output. [3 Marks]** 3 points

```
int main(){
    int p=8;
    int q=4;
    i=fork();
    if(i<0){
        printf("fork failed\n");
    }
    else if(i==0){
        p=p+q;
        q=p-q;
    }
    else{
        wait();
        j=fork();
        if(j<0){
            printf("fork failed\n");
        }
        else if(j==0){
            q=p*q;
            p=q/p;
        }
        else{
            wait();
            p=q-p;
            q=p+q;
        }
    }
    printf("value of p: %d\n",p);
    printf("value of q: %d\n",q);

    return 0;
}
```

8. [C01] 1.d) Answer the following question: **[3 Marks]**

3 points

In a significant research initiative addressing climate change, two pivotal groups play key roles: dedicated researchers who meticulously gather data from diverse sources, and skilled analysts responsible for employing advanced statistical models to derive actionable insights from this extensive dataset. To facilitate their collaboration, they have devised an effective approach for fluidly exchanging data and analysis instructions, thus enabling multiple analysts to work concurrently on this critical project.

Logically explain what type of communication method was used in the above given scenario.

message passing

9. [CO2] 2.a) Type the average waiting time and turnaround time in this form (as a response to this question) and show your calculation on paper : **[5 Marks]** 5 points

Draw a Gantt chart and illustrate the execution of the process using the **Round Robin** scheduling algorithm (**time quantum = 5 units**). Calculate the **average waiting** and **turnaround time**.

Process ID	Burst Time	Arrival Time	Priority
P1	11	5	1
P2	10	3	5
P3	5	9	4
P4	14	11	222222
P5	6	15	3

10. [CO2] 2.b) Answer the following question: **[2 Marks]** 2 points

Due to a calculation error, P4 has received an abnormally high priority value compared to other processes. Describe the problem this situation might cause by providing a scenario illustrating the issue, and suggest how to address it.

11. [CO3] 3.a) Type the answer in this form (as a response to this question) and show your calculation on paper : **[3 Marks]** 3 points

A system has processes to execute of which are 92% parallel. The number of cores currently available is 3. Calculate the number of cores required in order to increase the speedup approximately 2.25 times.

12. [CO3] 3.b) Answer the following question: **[2 Marks]** 2 points

Imagine you have a text editor that is running on multiple threads and has a python code execution feature. To achieve the code execution, the editor creates a child process and loads the python interpreter as a separate program. In this scenario, should the child process be a single-threaded or multi-threaded program? State your reasons.

13. [CO3] 3.c) Answer the following question: **[2 Marks]** 2 points

In a manufacturing facility, raw materials are received and undergo multiple processes, including quality control, production, and packaging. All these processes rely on a shared database to track inventory and production progress. **Identify** which parallelism technique can be applicable here?

task parallelism- different task on same data

Upload PDF

1. Upload your PDF in this form - [Click here](#)
2. SUBMIT both of the forms (this & PDF upload form)

Question: What measures can be taken at the operating system level to prevent data integrity problems caused by unauthorized access or malicious attacks?

Answer: Operating systems can implement access control mechanisms such as permissions and encryption to prevent unauthorized access to sensitive data. Additionally, security measures such as firewalls, intrusion detection systems, and regular security updates can help protect against malicious attacks that could compromise data integrity.

Question: What is a race condition in the context of operating systems?

Answer: A race condition occurs when the behavior of a system depends on the timing or sequence of events, and the outcome is unpredictable due to concurrent execution of multiple threads or processes accessing shared resources.

Question: How does a race condition arise in multitasking operating systems?

Answer: In multitasking operating systems, multiple processes or threads may access shared resources such as variables, files, or hardware devices concurrently. If proper synchronization mechanisms are not in place, race conditions can occur when these processes or threads access and modify shared resources simultaneously, leading to unexpected or erroneous behavior.

Question: What are the consequences of race conditions in operating systems?

Answer: Race conditions can lead to various issues such as data corruption, deadlocks, or inconsistent program behavior. These issues can be particularly challenging to debug and reproduce since they depend on the specific timing of events during execution. Proper synchronization techniques such as locks, semaphores, or mutexes are essential to mitigate race conditions and ensure the correct operation of concurrent programs.

Question: How can shared variables contribute to the occurrence of race conditions in operating systems?

Answer: Shared variables are accessible by multiple threads or processes concurrently. When these variables are accessed and modified without proper synchronization mechanisms, race conditions may arise. Concurrent access to shared variables can lead to inconsistent or unpredictable results due to interleaved execution of instructions by different threads or processes.

Question: Explain the difference between a race condition and a deadlock in operating systems.

Answer: A race condition occurs when the outcome of a system depends on the timing or sequence of events, leading to unpredictable behavior. In contrast, a deadlock occurs when two or more processes are unable to proceed because each is waiting for the other to release a resource, resulting in a stalemate situation where none of the processes can progress further.

Question: How can operating systems prevent race conditions from occurring?

Answer: Operating systems can prevent race conditions by implementing proper synchronization mechanisms such as locks, semaphores, or atomic operations. These mechanisms ensure that only one thread or process accesses a shared resource at a time, preventing concurrent modifications that could lead to race conditions. Additionally, operating systems may provide higher-level constructs like monitors or message passing to facilitate safe concurrent access to resources.

Question: How can mutual exclusion be achieved in solving the critical section problem?

Answer: Mutual exclusion can be achieved by implementing synchronization mechanisms such as locks, semaphores, or mutexes. These mechanisms ensure that only one process or thread can enter the critical section at a time, preventing simultaneous access by other processes or threads and avoiding potential data corruption or inconsistency.

Question: How do software and hardware solutions differ in addressing the critical section problem?

Answer: Software solutions for the critical section problem involve implementing synchronization mechanisms such as locks, semaphores, or monitors within the operating system or application code. Hardware solutions may involve special CPU instructions or hardware support for atomic operations, which can provide efficient mutual exclusion without the need for software-based synchronization primitives. Both software and hardware solutions aim to ensure mutual exclusion while satisfying the requirements of progress and bounded waiting.

Question: How does Peterson's solution guarantee mutual exclusion?

Answer: Peterson's solution guarantees mutual exclusion by having each process set its flag to indicate its intention to enter the critical section. If both flags are set, the process that is not in its critical section will wait until the other process completes its critical section. The turn variable determines which process has priority to enter the critical section next.

Question: What are the limitations of Peterson's solution?

Answer: Peterson's solution is limited to only two processes and may not be scalable for systems with more than two processes. Additionally, it relies on busy waiting, which can waste CPU cycles if a process is repeatedly denied access to its critical section. Finally, it does not address issues such as deadlock or starvation, which may arise in more complex systems.

Question: What is the purpose of the TEST_AND_SET instruction in operating systems?

Answer: The TEST_AND_SET instruction is used to atomically test and set the value of a memory location. It is commonly used in synchronization mechanisms to implement mutual exclusion, such as in Peterson's solution for the Critical Section Problem.

Question: How does TEST_AND_SET facilitate mutual exclusion?

Answer: TEST_AND_SET ensures mutual exclusion by allowing only one process at a time to successfully set a shared flag variable to indicate its intention to enter the critical section. Other processes attempting to set the flag while it is already set will be blocked until the flag is cleared.

Question: What are the drawbacks of using TEST_AND_SET for synchronization?

Answer: While TEST_AND_SET is effective for implementing mutual exclusion, it can lead to busy waiting, where processes repeatedly attempt to acquire the lock by continuously polling the flag. This can waste CPU cycles and reduce overall system efficiency. Additionally, TEST_AND_SET does not prevent starvation or deadlock, so additional mechanisms may be needed to address these issues in more complex systems.

Question: What is the purpose of the COMPARE_AND_SWAP() operation in operating systems?

Answer: The COMPARE_AND_SWAP() operation, also known as CAS, is used to atomically compare the value of a memory location with an expected value and, if they match, update the value of that location to a new value. It is commonly used in synchronization mechanisms to implement mutual exclusion and lock-free algorithms.

Question: What are the advantages of using COMPARE_AND_SWAP() over other synchronization techniques?

Answer: COMPARE_AND_SWAP() offers several advantages, including its ability to perform atomic operations without requiring locks, which can lead to better performance in highly concurrent systems. Additionally, it provides a mechanism for implementing lock-free algorithms, which can be more scalable and efficient than traditional locking approaches.

Question: What is the purpose of semaphores in operating systems?

Answer: Semaphores are synchronization primitives used to control access to shared resources and coordinate the execution of processes or threads in a concurrent system. They help prevent race conditions and ensure mutual exclusion and synchronization between multiple processes.

Question: How do binary semaphores differ from counting semaphores?

Answer: Binary semaphores, also known as mutex locks, have only two states: locked (1) and unlocked (0). They are typically used for mutual exclusion, allowing only one process or thread to access a resource at a time. Counting semaphores, on the other hand, can have multiple integer values and are used to control access to a finite number of identical resources.

Thread Sample Questions

1. How does an operating system manage threads within a process?

Answer: The operating system allocates resources such as CPU time and memory to threads within a process, schedules their execution, and ensures synchronization between threads when accessing shared resources.

2. What are the advantages of using threads in an operating system?

Answer: Threads allow for concurrent execution within a process, enabling better utilization of CPU resources, improved responsiveness, and enhanced performance through parallelism.

3. How does thread scheduling differ from process scheduling in an operating system?

Answer: Thread scheduling involves selecting which thread to execute next on the CPU within a single process, focusing on factors like priority, time slicing, and synchronization. Process scheduling, on the other hand, involves selecting which process to execute next, considering factors such as CPU and I/O bursts, and may involve more overhead.

4. What are some common thread synchronization mechanisms used in operating systems?

Answer: Common thread synchronization mechanisms include locks (mutexes), semaphores, condition variables, and barriers. These mechanisms help coordinate the execution of threads, prevent race conditions, and ensure proper access to shared resources.

5. How do threads contribute to improved responsiveness in an operating system?

Answer: Threads allow multiple tasks to run concurrently within a process, enabling the operating system to respond to user interactions or system events more quickly by executing parallel tasks simultaneously.

6. What role do threads play in enhancing resource utilization within an operating system?

Answer: Threads enable better utilization of CPU resources by allowing the operating system to switch between executing tasks when one thread is waiting for I/O operations or other blocking tasks, thereby keeping the CPU busy and maximizing its efficiency.

7. How do threads facilitate parallelism and increased throughput in an operating system?

Answer: Threads enable the execution of multiple tasks simultaneously within a single process, harnessing parallelism to perform computations or handle requests concurrently, leading to increased throughput and overall system performance.

8. In what ways do threads support modular and scalable software design within an operating system?

Answer: Threads enable modular design by allowing developers to encapsulate different functionalities or tasks within separate threads, promoting code reuse and easier maintenance. Additionally, threads facilitate scalability by distributing work across multiple threads, allowing software to efficiently utilize multi-core processors and scale with increasing computational demands.

9. How does data parallelism leverage multiple threads in an operating system?

Answer: Data parallelism involves dividing a task into smaller sub-tasks that operate on different pieces of data concurrently. In an operating system, multiple threads can be used to process distinct chunks of data simultaneously, exploiting parallelism to improve performance.

10. What are the benefits of employing data parallelism in operating system tasks?

Answer: Data parallelism allows for more efficient utilization of CPU resources by distributing computational workloads across multiple threads. This leads to faster execution times and improved throughput for tasks such as data processing, rendering, or scientific computations.

11. How does load balancing play a role in optimizing data parallelism within an operating system?

Answer: Load balancing ensures that computational workloads are evenly distributed among available threads or processing units, maximizing the utilization of system resources. In data parallelism, load balancing techniques help prevent bottlenecks and ensure that each thread receives a balanced amount of work.

12. What strategies can an operating system employ to implement data parallelism effectively?

Answer: Operating systems can implement data parallelism through techniques such as task decomposition, where a task is divided into smaller sub-tasks that can be executed concurrently by multiple threads. Additionally, synchronization mechanisms such as barriers or data dependency tracking may be used to coordinate the execution of parallel tasks and ensure correct results.

13. How does task parallelism differ from data parallelism in the context of operating system threads?

Answer: Task parallelism involves dividing a program or task into smaller independent tasks that can be executed concurrently by separate threads. Unlike data parallelism, where multiple threads operate on different pieces of data simultaneously, task parallelism focuses on parallelizing distinct tasks or operations.

14. What are the advantages of leveraging task parallelism in operating system thread management?

Answer: Task parallelism allows for better utilization of multi-core processors by executing independent tasks concurrently, leading to improved performance and throughput. It also enhances responsiveness as tasks can be executed in parallel, reducing overall execution time.

15. How does task dependency management impact the effectiveness of task parallelism in an operating system?

Answer: Task dependency management involves identifying relationships between tasks and ensuring that dependent tasks are executed in the correct order to maintain program correctness. In task parallelism, efficient management of task dependencies is crucial to avoid race conditions and ensure proper synchronization between threads.

16. What strategies can operating systems employ to implement task parallelism effectively?

Answer: Operating systems can implement task parallelism through techniques such as task scheduling, where independent tasks are assigned to available threads for concurrent execution. Additionally, synchronization mechanisms like locks, semaphores, or barriers may be used to coordinate the execution of tasks and manage dependencies.

17. What are the key differences between the user-level threading and kernel-level threading models in operating system multithreading?

Answer: In user-level threading, thread management is handled entirely by the user-space application without kernel involvement, offering lightweight and flexible threading. In contrast, kernel-level threading relies on the operating system kernel to manage threads, providing better support for multiprocessor systems and ensuring thread scheduling and synchronization at the kernel level.

18. How does the many-to-one threading model differ from the one-to-one threading model in operating system multithreading?

Answer: In the many-to-one threading model, multiple user-level threads are mapped to a single kernel-level thread, resulting in all threads being managed by a single kernel thread. Conversely, the one-to-one

threading model assigns each user-level thread directly to a kernel-level thread, offering more parallelism and scalability but potentially incurring higher overhead due to increased kernel resources.

19. What advantages does the many-to-many threading model offer compared to other multithreading models in operating systems?

Answer: The many-to-many threading model combines aspects of both one-to-one and many-to-one models, allowing for a flexible mapping of user-level threads to kernel-level threads. This model offers improved scalability and performance by leveraging both user-level and kernel-level threading, enabling better resource utilization and load balancing across multiple processors.

20. How does the two-level threading model address the limitations of traditional multithreading models in operating systems?

Answer: The two-level threading model introduces a hybrid approach where a user-level thread scheduler manages lightweight user threads, while a kernel-level thread scheduler oversees kernel-level threads. This model combines the flexibility of user-level threading with the scalability and robustness of kernel-level threading, aiming to provide efficient multithreading support while minimizing overhead.

Department of Computer Science and Engineering
MIDTERM EXAMINATION Summer 2022
CSE321: Operating Systems

Total Marks: 30

Time Allowed: 1 Hour 15 min

[Answer All Questions. Understanding the question is a part of the exam.]

CO1 CO2	1.	<p>a) What is multiprogramming? What are the requirements for multiprogramming?</p> <p>b) Explain User mode and Kernel mode execution with example.</p> <p>c) Define system call? Explain how system call works using System Call Interface.</p>	3 4 3																								
CO2	2.	<p>a) Though we can increase the hardware resources as much as we want by money, still why do we implement the thread? Explain how the thread accelerates the computing power with an example.</p> <p>b) A system has processes to execute of which 40% is parallel. If the number of cores is increased from 2 to 4, what will be the increase in performance?</p> <p>c) Find the output of the following code -</p> <pre>int main() { int id, i; printf("Start of main...\n"); id = fork(); if (id > 0) { printf("Parent section...\n"); } else if (id == 0) { printf("\n fork created...\n"); } else { printf("\n fork creation failed!!!\n"); } printf("Printing the numbers from 1 to 10\n"); for (i = 1; i <= 10; i++) printf("%d ", i); printf("\n"); printf("End of the main function...\n"); return 0; }</pre>	3 3 4																								
CO3	3.	<p>a) Find which CPU scheduling is more efficient between priority scheduling and Round Robin in terms of waiting time and context switch. Where quantum time is 4 (q=4).</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Process</th><th>Arrival time</th><th>Burst time</th><th>Priority</th></tr> </thead> <tbody> <tr> <td>P0</td><td>14</td><td>9</td><td>5</td></tr> <tr> <td>P1</td><td>5</td><td>10</td><td>2</td></tr> <tr> <td>P2</td><td>0</td><td>13</td><td>3</td></tr> <tr> <td>P3</td><td>9</td><td>8</td><td>1</td></tr> <tr> <td>P4</td><td>20</td><td>11</td><td>1</td></tr> </tbody> </table>	Process	Arrival time	Burst time	Priority	P0	14	9	5	P1	5	10	2	P2	0	13	3	P3	9	8	1	P4	20	11	1	8
Process	Arrival time	Burst time	Priority																								
P0	14	9	5																								
P1	5	10	2																								
P2	0	13	3																								
P3	9	8	1																								
P4	20	11	1																								

		b) What is starvation and how can it be solved?	2
--	--	---	---

Department of Computer Science and Engineering
Midterm Examination Fall 2022
CSE 321: Operating Systems
[Set A]

Duration: 1 Hour 10 Minutes

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
1. a) State time-sharing system with an example. [3]
CO1 b) Mention the roles of the system call interface. [2]

2. a) Distinguish between CPU scheduler and job scheduler. [3]
CO2 b) Find the output of the following code snippet. [4]

```
int main(){
    int id;
    static int x = 10;
    int y = 5;
    id = fork();
    if (id < 0){
        printf("fork failed\n");
    }
    else if(id == 0){
        printf("child started\n");
        printf("child finished\n");
    }
    else{
        wait(NULL);
        printf("parent started\n");
        x=x-2;
        y=y+5;
        printf("values of x: %d & y: %d\n", x, y);
        printf("parent finished\n");
    }
    x=x+5;
    y=y-5;
    printf("values of x: %d & y: %d\n", x, y);
    printf("terminating\n");

    return 0;
}
```

3. CO3 a) When is CPU scheduling required? [2]
- b) Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 11 units). Calculate the average waiting time and number of context switching. [3+2+1]

Processes	Arrival Time	Burst Time
P1	3	37
P2	12	17
P3	58	28
P4	59	21

P5	68	19
----	----	----

c) Consider the following set of processes with the length of the CPU-burst time given in milliseconds. Draw the Gantt Charts illustrating the execution of these processes using preemptive priority (the lowest number implies a higher priority). Calculate the average turnaround time for the below data set.

[3+2]

Processes	Priority	Arrival Time	Burst Time
P1	12	0	4
P2	8	1	2
P3	6	2	3
P4	2	3	5
P5	4	4	1
P6	1	5	4
P7	3	6	6

Department of Computer Science and Engineering
Midterm Examination Fall 2022
CSE 321: Operating Systems
[Set B]

Duration: 1 Hour

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
1. a) Explain dual-mode operation. [3]
CO1 b) Briefly explain any two services of the OS. [2]

2. a) Explain each process state with an example. [3]
CO2 b) Find the output of the following code snippet. [4]

```
int main(){
    int id;
    static int x = 10;
    int y = 5;
    id = fork();
    if (id < 0){
        printf("fork failed\n");
    }
    else if(id == 0){
        printf("child started\n");
        x=x+5;
        y=y-3;
        printf("values of x: %d & y: %d\n", x, y);
        printf("child finished\n");
    }
    else{
        wait(NULL);
        printf("parent started\n");
        printf("parent finished\n");
    }
    x=x+5;
    y=y-5;
    printf("values of x: %d & y: %d\n", x, y);
    printf("terminating\n");

    return 0;
}
```

3. a) "Multilevel-queue can prevent starvation problem"-Justify your answer. [2]
CO3

Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 12 units). Calculate the average waiting time and number of context switching. [3+2+1]

Processes	Arrival Time	Burst Time
P1	3	37
P2	12	17
P3	62	28
P4	63	21
P5	72	19

- b. Consider the following set of processes with the length of the CPU-burst time given in milliseconds. **Draw** the Gantt Charts illustrating the execution of these processes using preemptive priority (the highest number implies a higher priority). **Calculate** the **average turnaround** time for the below data set. [3+2 points] [3+2]

Processes	Priority	Arrival Time	Burst Time
P1	2	0	4
P2	4	1	2
P3	6	2	3
P4	10	3	5
P5	8	4	1
P6	12	5	4
P7	9	6	6

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid Term
Duration: 1 Hour 10 minutes

Semester: Spring 2022
Full Marks: 30

CSE 321: Operating Systems

Answer the following questions.
Figures in the right margin indicate marks.

-
- | | | |
|-----------|---|--|
| 1.
CO1 | <p>a) Explain how the two modes of the hardware enable the operating system to securely control user processes. [2]</p> <p>b) Explain the differences between Multiprogramming and Multiprocessing with examples. [2]</p> | |
| 2.
CO2 | <p>a) Describe what is the process control block, its contents, and how it is used. In particular, describe its role in context switching. [2+1]</p> <p>b) Find the output of the following code snippet. [3]</p> | |

```
int main()
{
    int pid1, pid2;
    pid1 = fork();
    if (pid1 == 0) {
        pid2=fork();
        if(pid2 == 0) printf("Hello!\n");
        else{
            wait(NULL);
            printf("World!\n");
        }
    } else {
        wait(NULL);
        printf("Missed Me?\n");
    }
    printf("Don' t miss me!\n");
    return 0;
}
```

- | | | |
|-----------|---|--|
| 3.
CO4 | <p>a) Suppose, in a system, you can use up to 4 processors for 40% of the applications, which means 40% of the applications can run in parallel. Calculate the speedup if you increase the number of processors from 1 to 4. [4]</p> <p>b) Remember that <i>pthread_create(tid, NULL, fn, arg)</i> creates a new thread that executes the function <i>fn</i> with the argument <i>arg</i>, and <i>pthread_join(tid, NULL)</i> let the current thread wait for the thread with id = <i>tid</i> to complete execution. With this information in mind, find all possible outputs of the following program.</p> | |
|-----------|---|--|

```

int[] matrix = {4, 6, 9, 2, 5, 3, 0, 1, 11, 13, -1, 7};
void main() {
    pthread_t t1, t2;
    printf("Printing partial sums of the array");
    pthread_create(&t1, NULL, sum, 0);
    pthread_create(&t2, NULL, sum, 4);
    pthread_join(t1, NULL);
    sum(8);
}
void sum(int startIndex) {
    int partialSum = 0;
    for (int i = startIndex; i < startIndex+4; i++) {
        partialSum += matrix[i];
    }
    printf("For index %d to %d = %d", startIndex, startIndex+4, partialSum);
    pthread_exit(0);
}

```

[4]

4. Consider the following processes with arrival time and burst time at a specific moment in the ready queue that needs to be scheduled.

CO3

Process	Arrival Time	Burst Time
P1	2	8
P2	7	3
P3	5	10
P4	5	6

- Apply Shortest Remaining Time First (SRTF) scheduling algorithm and show the following - [2+2+1]
 - Gantt Chart
 - Average Waiting Time & Average Turnaround Time
 - Number of Context Switching
- Apply Round Robin (RR) scheduling algorithm with quantum = 3 and show the following - [2+2+1]
 - Gantt Chart
 - Average Waiting Time & Average Turnaround Time
 - Number of Context Switching
- Find the best-suited algorithm between these two and give your reasoning. [2]

Department of Computer Science and Engineering
Midterm Examination Spring 2023
CSE 321: Operating Systems



Duration: 1 Hour 15 Minutes

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
- | | | |
|-----|---|-----|
| 1. | a) Explain why main memories are usually volatile. | [2] |
| CO1 | b) Distinguish between monolithic and microkernel OS structures. | [3] |
| | c) A program has a process that will allow its child process to complete first. Which system call can be used in this scenario? Explain what may happen in absence of this system call. | [2] |
| | d) Find the output of the following code snippet. | [3] |

```
const int len = 2;
int main(){
    int id;
    int a[] = {10, 7};
    int b = len-1;
    id = fork();
    if (id < 0){
        printf("fork failed\n");
    }
    else if(id == 0){
        printf("child process executing\n");
    }
    else{
        wait(NULL);
        printf("parent process executing\n");
        a[b-1]=a[b-1]-2;
        a[b] = a[b]+2;
    }
    for(int i=0; i<len; i++){
        printf("value of a[%d]: %d\n", i, a[i]);
    }
    return 0;
}
```

2.
CO2

Processes	Arrival Time	Burst Time
P1	0	11
P2	20	6
P3	14	9
P4	20	8
P5	15	8
P6	16	8
P7	2	2

- a) Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 5 units). Calculate the average waiting and turnaround time, [3+2]
- b) Apply Shortest Remaining Time First (SRTF) scheduling algorithm. Draw the Gantt chart and Calculate the average waiting and turnaround time. [2+2]
- c) Compare the results and identify the most suitable scheduling algorithm in this scenario. [1]

3.
CO3

- a) A system has processes to execute of which 35% is parallel. If the number of cores is increased from 3 to 5, Explain what will be the increase/decrease in performance? [2]
- b) A program has multiple threads that need to perform a task which involves similar computation on a large scale of data. Identify which kind of parallelism can be applied in this scenario. Provide proper justification with a real-life example of the scenario mentioned above. [3]

Department of Computer Science and Engineering
Midterm Examination Spring 2023
CSE 321: Operating Systems

B

Duration: 1 Hour 15 Minutes

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
1. CO1
- a) Briefly explain dual mode operation of OS. [2]
 - b) Distinguish between layered and microkernel OS structures. [3]
 - c) Two processes (A and B) are distributed across multiple physical machines or networked systems. Which technique can be used to achieve inter-process communication in this scenario? Is it suitable for exchanging large amounts of data? Provide proper justification to support your answer. [2]
 - d) Find the output of the following code snippet. [3]

```
const int len = 2;
int main(){
    int id;
    int a[] = {5, 8};
    int b = len-1;
    id = fork();
    if (id < 0){
        printf("fork failed\n");
    }
    else if(id > 0){
        wait(NULL);
        printf("parent process executing\n");
    }
    else{
        printf("child process executing\n");
        a[b-1]=a[b-1]+2;
        a[b] = a[b]-3;
    }
    for(int i=0; i<len; i++){
        printf("value of a[%d]: %d\n", i, a[i]);
    }
    return 0;
}
```

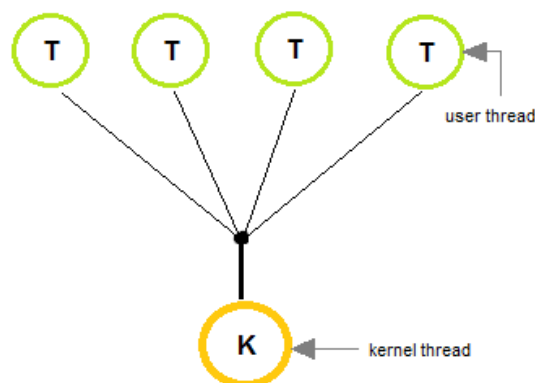
2.
CO2

Processes	Arrival Time	Burst Time
P1	0	11
P2	20	6
P3	14	9
P4	20	8
P5	15	8
P6	16	8
P7	2	2

- a) Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 5 units). Calculate the average waiting and turnaround time, [3+2]
- b) Apply Shortest Remaining Time First (SRTF) scheduling algorithm. Draw the Gantt chart and Calculate the average waiting and turnaround time. [2+2]
- c) Compare the results and identify the most suitable scheduling algorithm in this scenario. [1]

3.
CO3

- a) A system has processes to execute of which 45% is parallel. If the number of cores is increased from 2 to 4, Explain what will be the increase/decrease in performance. [2]
- b) Describe the multithreading model which is shown in the picture below. Identify the issue in this model and suggest another multithreading model which is free from that issue. [3]



Department of Computer Science and Engineering
Midterm Examination Summer 2023
CSE 321: Operating Systems



Duration: 1 Hour 15 Minutes

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
1. a) Why do we need cooperating processes, and what are the two models of interprocess communication? Additionally, mention the strengths and weaknesses of each approach? [3]
CO1
- b) Discuss one drawback faced by multi-programmed OS architecture and suggest possible way(s) to overcome it. [2]
- c) What is the purpose of system calls? Which of the following instructions should be privileged? [1+1]
i. Set value of timer ii. Read the clock iii. Clear memory iv. Issue a trap instruction.
v. Turn off interrupts vi. Modify entries in device-status table
- d) Find the output of the following code snippet. Your output should exactly match with the original output. [3]

```
int main() {
    pid_t child_pid;
    int global_a = 90, b = 11;
    char message[] = "Hello, from the ";
    printf("Parent process started\n");
    child_pid = fork();
    if (child_pid == -1) {
        printf("Fork Failed\n");
    } else if (child_pid > 0) {
        wait(NULL);
        global_a += 97;
        printf("%sAddition: %d + %d = %d\n", message,
        global_a, b, global_a);
        printf("%sSubtraction: %d - %d = %d\n", message, b,
        global_a, b);
    } else {
        b *= 7;
        printf("Multiplication: %d * %d = %d\n", global_a,
        b, global_a);
        printf("Division: %d / %d = %d\n", b, global_a, b);
    }
    return 0;
}
```

2.
CO2

Processes	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	6	4	1
P3	7	6	6
P4	7	1	3
P5	8	7	4
P6	19	7	5

a) Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 5 units). Calculate the average waiting and turnaround time. [3+2]

b) Apply Preemptive Priority scheduling algorithm. Draw the Gantt chart and Calculate the average waiting and turnaround time. [2+2]

c) Compare the results and identify the most suitable scheduling algorithm in this scenario. [1]

3.
CO3

a) Explain data parallelism with an example. [1.5]

b) You are developing a real-time embedded system for a safety-critical application, such as an advanced driver assistance system (ADAS) for autonomous vehicles. The system's primary goal is to process sensor data, make critical decisions, and take actions in real-time to ensure the safety of passengers and pedestrians. However, you may assume there's no limitation of computational resources on the project you are working on. [1.5]

Based on the scenario, which multi-threading model would you recommend for implementation? Provide necessary justification.

c) A system has processes to execute of which 30% is serial. If the number of cores is decreased from 9 to 4, Explain the change in the performance. [2]

Department of Computer Science and Engineering
Midterm Examination Summer 2023
CSE 321: Operating Systems

B

Duration: 1 Hour 15 Minutes

Total Marks: 25

Answer the following questions.
Figures in the right margin indicate marks.

-
1. a) Define the term "interrupt" in the context of operating systems. Mention one advantages and one disadvantage of a multiprocessor system. [3]
CO1
- b) A process from its creation till its completion will go through various states. To enter different states, the process requires the decision of different types of scheduler. State the name of different schedulers for different process states with justification. [2]
- c) What is the purpose of dual mode operation? Which of the following instructions should be privileged? [1+1]
i. Access I/O device ii. Set value of timer iii. Read the date in the calendar.
iv. Clear memory v. Switch from user to kernel mode vi. Turn off interrupts.
- d) Find the output of the following code snippet. Your output should exactly match with the original output. [3]

```
int main() {
    pid_t child_pid;
    int global_a = 68, b = 10;
    char message[] = "Hello, from the ";
    printf("Parent process started\n");
    child_pid = fork();
    if (child_pid == -1) {
        printf("Fork Failed\n");
    } else if (child_pid > 0) {
        wait(NULL);
        b *= 38;
        printf("Multiplication: %d * %d = %d\n", global_a,
b, global_a);
        printf("Division: %d / %d = %d\n", b, global_a, b);
    } else {
        global_a += 98;
        printf("%sAddition: %d + %d = %d\n", message,
global_a, b, global_a);
        printf("%sSubtraction: %d - %d = %d\n", message, b,
global_a, b);
    }
    return 0;
}
```

2.
CO2

Processes	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	6	4	1
P3	7	6	6
P4	7	1	3
P5	8	7	4
P6	19	7	5

a) Draw a Gantt chart and illustrate the execution of the process using the Round Robin scheduling algorithm (time quantum = 5 units). Calculate the average waiting and turnaround time. [3+2]

b) Apply Preemptive Priority scheduling algorithm. Draw the Gantt chart and Calculate the average waiting and turnaround time. [2+2]

c) Compare the results and identify the most suitable scheduling algorithm in this scenario. [1]

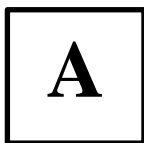
3.
CO3

a) Explain task parallelism with an example. [1.5]

b) You are developing a lightweight, user-level threading library for a resource-constrained embedded system. The embedded system has limited processing power and memory, and it does not provide native support for multithreading at the kernel level. The primary goal is to allow concurrent execution of multiple tasks while minimizing the overhead of managing threads. [1.5]

Based on the scenario, which multi-threading model would you recommend for implementation? Provide necessary justification.

c) A system has processes to execute of which 32% is serial. If the number of cores is decreased from 8 to 2, Explain the change in the performance. [2]




Department of Computer Science and Engineering
Final Examination Fall 2022
CSE 321: Operating Systems

Duration: 2 Hours

Total Marks: 40

Answer the following questions.
 Figures in the right margin indicate marks.

<div>1. CO4</div>	<div>a) A system has processes to execute of which 50% is parallel. If the number of cores is increased from 1 to 4, what will be the increase in performance?</div> <div>b) Distinguish between many-to-many and many-to-one multithreading models.</div>	<div>[2]</div> <div>[2]</div>				
<div>2. CO5</div>	<div>a) Suppose a faculty member can take a maximum of 5 groups of students for doing a thesis under him in a semester. In a particular semester, a total of 9 groups applied for doing a thesis under his supervision. Among them, he selected 5 groups and kept the rest of the other groups on a waiting list for the next semester where groups will be selected according to a first come first serve manner from the waiting list if any of his slots gets free. Logically explain which synchronization method has been used here.</div> <div>b) For Peterson’s problem below conditions will be applied.<ul style="list-style-type: none">• There are two processes: P1 and P2.• Each Statement takes 3ms to execute.• Context Switch will occur after 9ms.• Both the Critical & Remainder section contains 3 statements.• For P1: i=0 and j=1• For P2: i=1 and j=0• turn=0• flag[0] = FALSE, flag[1] = TRUE</div> <div>The structure of process P_i in Peterson’s solution:<div><div>do{ flag[i] = true; turn = j; while(flag[j] == true && turn == 1){ //busy wait } //critical section flag[i] = false; //remainder section }while(true);</div></div></div>	<div>[2]</div> <div>[4]</div>				
<div>Complete the table given below for processes P1 and P2 using Peterson’s solution.</div> <table><tr><td>Process 1: i=0, j=1</td><td>Process 2: i=1, j=0</td></tr><tr><td></td><td></td></tr></table>			Process 1: i=0, j=1	Process 2: i=1, j=0		
Process 1: i=0, j=1	Process 2: i=1, j=0					

Department of Computer Science and Engineering
Final Examination Fall 2022
CSE 321: Operating Systems

B

Duration: 2 Hours

Total Marks: 40

Answer the following questions.
Figures in the right margin indicate marks.

-
1. **CO4** a) A system has processes to execute of which **30%** is serial. If the number of cores is increased from **1** to **3**, what will be the **increase** in performance? [2]
b) **Distinguish** between **many-to-one** and **one-to-one** multithreading models. [2]
2. **CO5** a) Suppose a medical center is providing Covid vaccination. In that center maximum of 6 people can take vaccines at a time in separate booths. But approximately 50 people went there to take vaccines on a particular day. Therefore, the authorities have decided that they will provide vaccines to 6 people at a time and keep others waiting in a queue. If any of the vaccine booths get free a person from the queue will be taken to that booth according to the first come first serve manner for vaccination. **Logically explain** which synchronization method has been used here [2]
b) For Peterson's problem below conditions will be applied.
 - There are two processes: P1 and P2.
 - Each Statement takes 4ms to execute.
 - Context Switch will occur after 12ms.
 - Both the Critical & Remainder section contains 3 statements.
 - For P1: i=0 and j=1
 - For P2: i=1 and j=0
 - turn=0
 - flag[0] = FALSE, flag[1] = FALSE


The structure of process P_i in Peterson's solution:

```
do{
    flag[i] = true;
    turn = j;
    while(flag[j] == true && turn == 1){
        //busy wait
    }
    //critical section
    flag[i] = false;
    //remainder section
}while(true);
```

[4]

Complete the table given below for processes P1 and P2 using **Peterson's solution**.

Process 1: i=0, j=1	Process 2: i=1, j=0

BRAC UNIVERSITY

Department of Computer Science and Engineering

Examination: Final
Duration: 2 Hours

Semester: Spring 2022
Full Marks: 40

CSE 321: Operating Systems

Answer the following questions.
Figures in the right margin indicate marks.

-
1. CO5
- a) Explain Race conditions with an example. Mention how we protect the system from this phenomenon. [2+1]
- b) Explain with a code example how a careless ordering of semaphore operations can lead to a deadlock situation among two processes. [2]
- c) Suppose, you have to design an online consultation system for the teachers and students of your university. There are certain constraints that you have to keep in mind while designing the system - [5]
- i. the teachers can set their status whether they are available to give consultation or not. A teacher will set him unavailable after the giveConsultation() function.
 - ii. the students will enter a voice channel if the teacher is available for consultation.
 - iii. the students will wait if one student is in consultation with the teacher

Now, you have to design the teacher and student function using semaphores so that synchronization can be achieved among them maintaining the constraints mentioned above. You can use the following code template given below and complete it. Mention the initial semaphore values before writing the functions.

```
//initialize the semaphore values here
S=1
teacher(){
    // write semaphore code here
    giveConsultation(s);

    // write semaphore code here
}

student(){
    // write semaphore code here
    takeConsultation(s);
```

```
        // write semaphore code here  
    }
```

```
Take consultation(s,std):  
s-=1  
if s<0:  
std.append(list)  
else:  
give consultation(s, std)
```

```
Give consultation(s,std):  
S++  
If s>=0:  
Wakeup(std)
```

[3]

[2]

[4]

[2]

[4]

Department of Computer Science and Engineering
Final Examination Spring 2023
CSE 321: Operating Systems

A

Duration: 2 Hours

Total Marks: 40

Answer the following questions.
 Figures in the right margin indicate marks.


- 1.** a) In a restaurant, there are 3 washrooms available for male guests and 1 washroom available for female guests. There is a strict rule maintained by the authorities that neither men can use the female washroom nor women can use any of the male washrooms. On a random day during lunch time, 9 male guests needed to use washroom facilities at the very same time. But none of them were allowed to use the female washroom although it was vacant. Therefore, only 3 guests were able to get access to washrooms at a time and others had to wait while maintaining a queue. If any of the washrooms gets vacant, a person from the queue can get access to that. Logically **explain** which synchronization method has been used here. [3]
- CO4**
- b) For Peterson's problem below conditions will be applied.
- There are two processes: P_1 and P_2 .
 - Each Statement takes 4ms to execute, P_1 gets executed first
 - Context Switch will occur after 16ms.
 - Critical section contains 4 statements.
 - Remainder section contains 2 statements.
 - For P_1 : $i = 0$ and $j = 1$
 - For P_2 : $i = 1$ and $j = 0$
 - $turn = 0$
 - $flag[0] = \text{FALSE}$, $flag[1] = \text{TRUE}$

The structure of process P_i in Peterson's solution:

```
do{
    flag[i] = true;
    turn = j;
    while(flag[j] == true && turn == 1){
        //busy wait
    }
    //critical section
    flag[i] = false;
    //remainder section
}while(true);
```

Complete the table given below for processes P_1 and P_2 using Peterson's solution.

[4]

Process 1: $i = 0, j = 1$	Process 2: $i = 1, j = 0$
	

--	--

Department of Computer Science and Engineering
Final Examination Spring 2023
CSE 321: Operating Systems

B

Duration: 2 Hours

Total Marks: 40

Answer the following questions.
 Figures in the right margin indicate marks.

1.
CO4

- a) In a restaurant, there are 3 washrooms available for male guests and 1 washroom available for female guests. There is a strict rule maintained by the authorities that neither men can use the female washroom nor women can use any of the male washrooms. On a random day during lunchtime, 4 female guests needed to use washroom facilities at the very same time. But none of them were allowed to use any of the male washrooms although two of them were vacant. Therefore, only 1 guest was able to get access to the washroom at a time and others had to wait while maintaining a queue. If the washroom gets vacant, a guest from the queue can get access to that. Logically **explain** which synchronization method has been used here.
- = semaphore + queue

[3]

b) For Peterson's problem below conditions will be applied.

- There are two processes: P_1 and P_2 .
- Each Statement takes 5ms to execute, P_1 gets executed first
- Context Switch will occur after 20ms.
- Critical section contains 2 statements.
- Remainder section contains 4 statements.
- For P_1 : $i = 0$ and $j = 1$
- For P_2 : $i = 1$ and $j = 0$
- $turn = 0$
- $flag[0] = \text{FALSE}$, $flag[1] = \text{TRUE}$

The structure of process P_i in Peterson's solution:

```
do{
    flag[i] = true;
    turn = j;
    while(flag[j] == true && turn == 1){
        //busy wait
    }
    //critical section
    flag[i] = false;
    //remainder section
}while(true);
```

Complete the table given below for processes P_1 and P_2 using Peterson's solution.

[4]

Process 1: $i = 0, j = 1$	Process 2: $i = 1, j = 0$
$F[0]=T, T=1$ While T	
	$F[1]=T, T=0$

While F, CS1	
While T	
	CS2, F[1]=F, RS1-2
While F CS 1-2, F[0]=F	
	RS 3-4
RS1-4	

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Semester Final
Duration: 2 Hours 30 minutes

Semester :Summer 2019
Full Marks:64

CSE 321: Operating Systems

Answer the following questions.
Figures in the right margin indicate marks.

Name:	ID:	Section:
-------	-----	----------

- 6.** **List** the optimization criteria of CPU scheduling? 1
CO3 The SJF CPU scheduling technique preempts an executing process. Using the SJF +
policy, **construct** a Gantt Chart and **compute** the waiting time for the 6 processes 4
tabulated below (time in milliseconds): +
3

Process	Burst Time	Arrival Time
P1	8	3
P2	12	10
P3	8	13
P4	6	14
P5	7	19
P6	8	88

Department of Computer Science and Engineering
Final Examination Summer 2023
CSE 321: Operating Systems



Duration: 1 Hour 50 Minutes

Total Marks: 40

Answer the following questions.
 Figures in the right margin indicate marks.

1. **CO4** a) Imagine a computer lab with multiple computers, and equipped with printers. Students use these computers for various tasks and may want to print documents. However, there are only a limited number of printers available. How do you solve this issue using semaphore? Your answer should have the steps associated in solving the given scenario. [3]

b) What is meant by busy waiting in the implementation of a critical section solution, and why is it considered problematic? [1.5]

- c) For Peterson's problem below conditions will be applied.
- There are two processes: P1 and P2. P2 gets to execute first.
 - Each Statement takes 4ms to execute.
 - Context Switch will occur after 12ms.
 - Critical section contains 4 statements.
 - Remainder section contains 3 statements.
 - For P1: i=0 and j=1
 - For P2: i=1 and j=0
 - turn=0
 - flag[0] = FALSE, flag[1] = TRUE


The structure of process P_i in Peterson's solution:

```
do{
    flag[i] = true;
    turn = j;
    while(flag[j] == true && turn == 1){
        //busy wait
    }
    //critical section
    flag[i] = false;
    //remainder section
}while(true);
```

Complete the table below for processes P₁ and P₂ using Peterson's solution.

[3.5]

Process 1: i = 0, j = 1	Process 2: i = 1, j = 0
	Flag[1]=T T=0 While False
Flag[0]=T T=1 While T	

	CS 1-3
While T	
	CS4, Flag[1]=False RS1
While F CS 1-2	
	RS 2,3
Rest of it	
	

Department of Computer Science and Engineering
Final Examination Summer 2023
CSE 321: Operating Systems

B

Duration: 1 Hour 50 Minutes

Total Marks: 40

Answer the following questions.
 Figures in the right margin indicate marks.

1. **a)** Imagine a computer lab with multiple computers, and equipped with printers. Students use these computers for various tasks and may want to print documents. However, there are only a limited number of printers available. How do you solve this issue using semaphore? Your answer should have the steps associated in solving the given scenario. [3]

CO4

b) How do semaphores and mutexes differ in their implementation. [1.5]

- c)** For Peterson's problem below conditions will be applied.
- There are two processes: P1 and P2. P2 gets to execute first.
 - Each Statement takes 4ms to execute.
 - Context Switch will occur after 12ms.
 - Critical section contains 4 statements.
 - Remainder section contains 3 statements.
 - For P1: i=0 and j=1
 - For P2: i=1 and j=0
 - turn=0
 - flag[0] = FALSE, flag[1] = TRUE

The structure of process Pi in Peterson's solution:

```
do{
    flag[i] = true;
    turn = j;
    while(flag[j] == true && turn == 1){
        //busy wait
    }
    //critical section
    flag[i] = false;
    //remainder section
}while(true);
```

Complete the table below for processes P₁ and P₂ using **Peterson's solution**. [3.5]

Process 1: i = 0, j = 1	Process 2: i = 1, j = 0
	f-1=T t=0 while F
