Section A [26 marks]

1. [1 mark]
Which of the following is the most efficient for a multiple-word I/O transfer.
A) Interrupt-driven I/O
B) Programmed I/O
C) Direct memory access
D) All the above are equally efficient

Answer: C

2a. [3 marks]
Complete the following table by inserting "high", "middle" or "low".

Type of memory	Cost per bit	Access time
Main memory	2(i)	2(iv)
Cache	2(ii)	2(v)
Disk	2(iii)	2(vi)

Answer:

Type of memory	Cost per bit	Access time
Main memory	**Middle**	**Middle**
Cache	**High**	**Low**
Disk	**Low**	**High**

2b. [3 marks]
Complete the following table by inserting "high", "middle" or "low".

Type of memory	Capacity	Access frequency by processor
Main memory	2(i)	2(iv)
Cache	2(ii)	2(v)
Disk	2(iii)	2(vi)

Answer:

Type of memory	Capacity	Access frequency by processor
Main memory	**Middle**	**Middle**
Cache	**Low**	**High**
Disk	**High**	**Low**

3a. [1 mark]
The collection of program, data, stack, and attributes is referred to as the _____ .

3b. [1 mark]
The _____ is the collection of program, data, stack, and attributes defined in the process control block.

Answer: process image

4. [1 mark]
The requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources is _____ .

Answer: mutual exclusion

5a. [1 mark]
A semaphore that does not specify the order in which processes are removed from the queue is a _____ semaphore.

Answer: weak

5b. [1 mark]
A semaphore whose definition includes the policy that the process that has been blocked the longest is released from the queue first is called a _____ semaphore.

Answer: strong

6a. [1 mark]
A _____ occurs when multiple processes or threads read and write data items so that the final result depends on the order of execution of instructions in the multiple processes.

6b. [1 mark]
A situation in which multiple threads or processes read and write a shared data item and the final result depends on the relative timing of their execution is a _____ .

Answer: race condition

7. [1 mark]
Probably the most useful combination, _____ allows a process to send one or more messages to a variety of destinations as quickly as possible.
A) blocking send, blocking receive
B) non-blocking send, blocking receive
C) non-blocking send, non-blocking receive
D) blocking send, non-blocking receive

Answer: B

8. [1 mark]
A situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something is a _____ .

Answer: deadlock

9. [1 mark]
A _____ is a semaphore that takes on only the values of 0 and 1.

Answer: binary semaphore

10. [1 mark]
_____ is a code segment within a process that requires access to shared resources and that must not be executed while another process is in a corresponding code segment.

Answer: Critical section

11a. [1 mark]
_____ refers to the ability of an OS to support multiple, concurrent paths of execution within a single process.

11b. [1 mark]
_____ is a technique in which a process, executing an application, is divided into threads that can run concurrently.

Answer: Multithreading

12a. [2 marks for correct order, 1 mark for 1 or 2 events not in order]
After the I/O device issues an interrupt signal to the processor, the following events will happen. Write down the correct sequence of those events.

(i) Processor loads new PC value based on interrupt
(ii) Processor finishes execution of current instruction
(iii) Processor pushes PSW and PC onto control stack
(iv) Processor restores PSW and PC from the control stack
(v) Process the interrupt

12b. [2 marks for correct order, 1 mark for 1 or 2 events not in order]
After the I/O device issues an interrupt signal to the processor, the following events will happen. Write down the correct sequence of those events.

(i) Processor loads new PC value based on interrupt
(ii) Processor finishes execution of current instruction
(iii) Processor restores PSW and PC from control stack
(iv) Processor pushes PSW and PC onto the control stack
(v) Process the interrupt

Answer: (ii), (iv), (i), (v), (iii)

13. [2 marks]

Which of the following statements are correct description of kernel-level threads (KLTs)?

(i) KLTs are created by invoking an application-level function.
(ii) The kernel is aware of the existence of KLTs
(iii) The transfer of control from one thread to another within the same process requires a mode switch to the kernel is one of the advantages of the KLT approach.
(iv) Using KLTs cannot provide a better performance than a single-threaded solution on a single-processor system.

A) (ii), (iii) and (iv)
B) (i) only
C) (ii) only
D) (ii) and (iv)

Answer: C

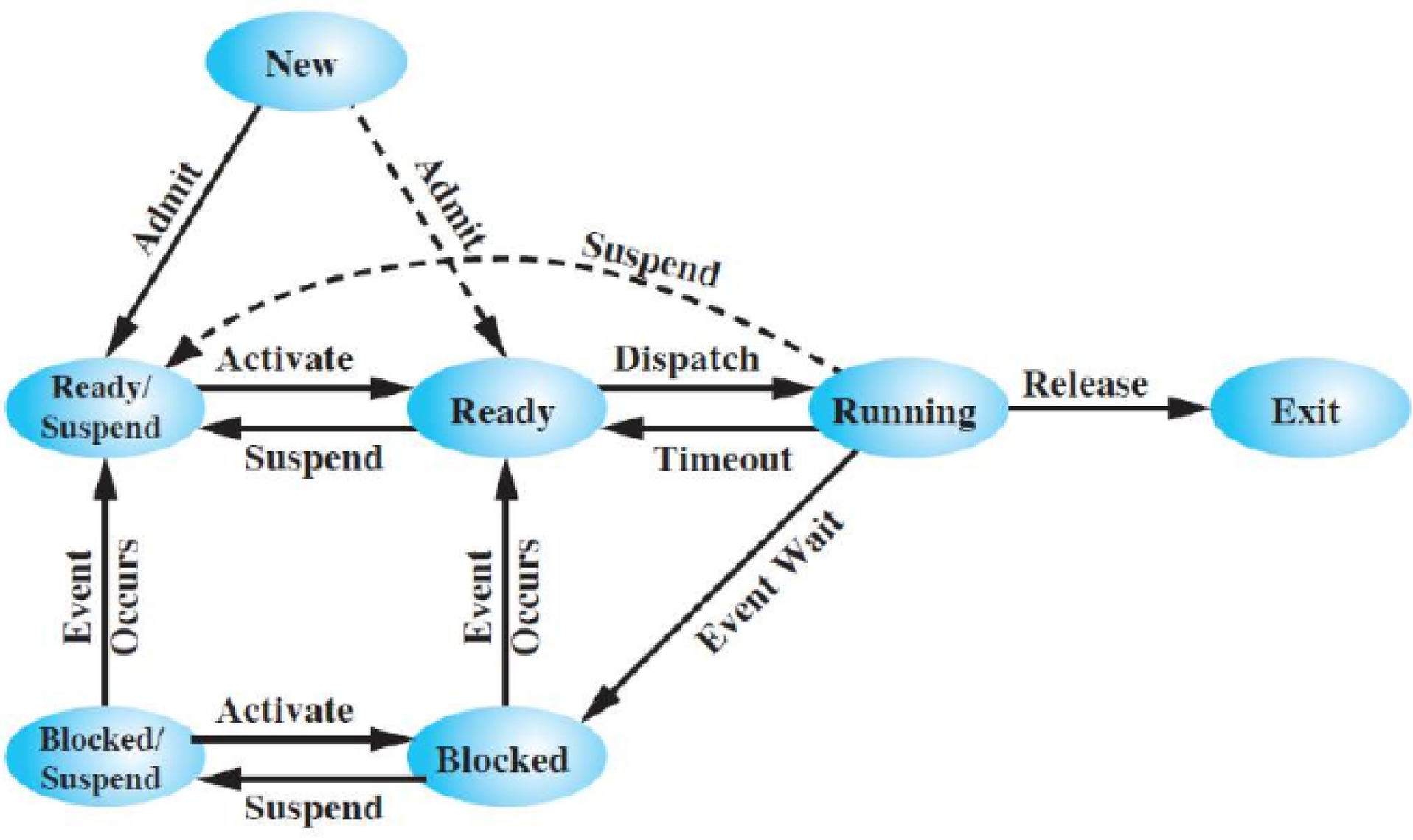
14. [3 marks]
Complete the following table by indicating whether the state is for process only, thread only, or both.

State	process/thread/both
Running	14(i)
Suspend	14(ii)
Blocked	14(iii)

Answer:

State	process/thread/both
Running	**Both**
Suspend	**Process**
Blocked	**Both**

Consider the following 7-state process model in answering questions 15 and 16.



15a. [3 marks]
Complete the following table by inserting the state of a process.

State	Status of the process	Location of the process
15(i)	Available for execution	Main memory
15(ii)_	Completed running	Main memory
15(iii)	Awaiting an event	Secondary memory

Answer:

State	Status of the process	Location of the process
Ready	Available for execution	Main memory
Exit	Completed running	Main memory
Blocked/Suspend	Awaiting an event	Secondary memory

15b. [3 marks]
Complete the following table by inserting the state of a process.

State	Status of the process	Location of the process
15(i)	Available for execution	Secondary memory
15(ii)_	Completed running	Main memory
15(iii)	Awaiting an event	Main memory

Answer:

State	Status of the process	Location of the process
Ready/Suspend	Available for execution	Secondary memory
Exit	Completed running	Main memory
Blocked	Awaiting an event	Main memory

16a. [3 marks]
Complete the table by filling the state transition (may or may not be shown on the diagram). The first one is an example.

Event	Transition
The dispatcher chooses a process to run.	Ready → Running
A process requests a resource which is not currently available.	16(i)
OS swaps out a periodic process waiting for the next execution.	16(ii)
A resource for which a process waiting in the main memory becomes available.	16(iii)

Answer:

Event	Transition
The dispatcher chooses a process to run.	Ready → Running
A process requests a resource which is not currently available.	**Running → Blocked**
OS swaps out a periodic process waiting for the next execution.	**Blocked→ Blocked/Suspend**
A resource for which a process waiting in the main memory becomes available.	**Blocked → Ready**

16b. [3 marks]
Complete the table by filling the state transition (may or may not be shown on the diagram).
The first one is an example.

Event	Transition
The dispatcher chooses a process to run.	Ready → Running
OS brings in a high-priority ready process from disk into memory.	16(i)
A parent process terminates its child process waiting for execution.	16(ii)
A process has reached the maximum allowable processor time.	16(iii)

Answer:

Event	Transition
The dispatcher chooses a process to run.	Ready → Running
OS brings in a high-priority ready process from disk into memory.	**Ready/suspend → Ready**
A parent process terminates its child process waiting for execution.	**Ready → Exit**
A process has reached the maximum allowable processor time.	**Running→Ready**

Section B [24 marks]

Q1a/b/c/d [8 marks]
Consider the following program fragment.

```
int x = 0/1/2/3;

void *runner (void *arg) {
      x+=2;
      cout << "thread: " << x << endl;
      pthread_exit(NULL);
}

int main ()
{
      int pid;
      pthread_t tid;

      pid = fork();
      x++;
      if (pid == 0) {
            pthread_create(&tid, NULL, runner, NULL);
            pthread_join(tid, NULL);
            x+=3;
            cout << "process1: " << x << endl;
      }
      else {
            wait(NULL);
            x+=4;
            cout << "process2: " << x << endl;
      }
}
```

(i) [4 marks]

List the outputs of the program. Show the order and values clearly.

[4 marks (3 marks for correct values and 1 mark for correct order)]

x=0	x=1	x=2	x=3
thread: 3 process1: 6 process2: 5	thread: 4 process1: 7 process2: 6	thread: 5 process1: 8 process2: 7	thread: 6 process1: 9 process2: 8

(ii) [4 marks]

Briefly explain the values of the above output. No need to explain the order.

- **When fork is called, a copy of the process image of the parent (including the variable x) is made for the child. So, the value of x in the parent (in the else statement) is not affected by the child.**
- **In contrast, the child and the thread created by the child have access to the same variable x. So, they can see the changes made by each of other.**

Q2a [4 marks]

Refer to the following solution to the bounded-buffer producer/consumer problem using semaphore. Assume that buffer size is 10.

```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1, n= 0, e= sizeofbuffer;
void producer()
{
     while (true) {
          produce();
          semWait(e);
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          semSignal(e);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

(i) What is the value of each of the three semaphores when a producer is appending data into an empty buffer while no consumer is waiting?

s=0, n=0, e=9

(ii) State the role of the semaphore s?

To enforce mutual exclusion.

Q2b [4 marks]

Refer to the following solution to the bounded-buffer producer/consumer problem using semaphore. Assume that buffer size is 10.

```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1, n= 0, e= sizeofbuffer;
void producer()
{
     while (true) {
          produce();
          semWait(e);
          semWait(s);
          append();
          semSignal(s);
          semSignal(n);
     }
}
void consumer()
{
     while (true) {
          semWait(n);
          semWait(s);
          take();
          semSignal(s);
          semSignal(e);
          consume();
     }
}
void main()
{
     parbegin (producer, consumer);
}
```

(i) What is the value of each of the three semaphores when a consumer is taking data from a full buffer while no producer is waiting?

s=0, n=9, e=0

(ii) State the role of the semaphore s?

To enforce mutual exclusion.

Q3 [12 marks]

Consider the following solution using ***semaphores*** to the *one-writer many-readers* problem.

```
int         readcount;
Semaphore   sem_x, sem_y;
```

Writer

semWait(sem_x);
/* writing performed */
semSignal(sem_x);

Readers

semWait(sem_y);
readcount++;
if readcount==1 then semWait(sem_x);
semSignal(sem_y);
/* reading performed */
semWait(sem_y);
readcount--;
if readcount==0 then semSignal(sem_x);
semSignal(sem_y);

A) [7 marks]

Suppose a *reader* is now reading in the system. No other readers or writer are waiting for the time being.

(i) [3 marks]

What are the current values of the three variables?

- **readcount = 1**
- **sem_x = 0**
- **sem_y = 1**

(ii) [2 marks]

What will happen when the *writer* wants to write while the *first reader* is still reading?

- **The value of sem_x will be changed to -1**
- **The writer will be blocked on sem_x**

(iii) [2 marks]

Following the *writer*, what will happen when a *second reader* wants to read while the *first reader* is still reading?

- **The value of readcount will be incremented to 2**
- **The second reader can read without waiting**

B) [2 marks]

Which semaphore can be replaced by a ***mutex***? Explain

- **sem_y can be replaced by a mutex because it is the same process (reader) that locks and unlocks the mutex.**

C) [3 marks]

Explain the potential issue/problem with this solution.

- **In this solution, readers have higher priority.**
- **Once a single reader has begun to access the data area, it is possible for readers to retain control of the data area as long as there is at least one reader reading. Therefore, writers are subject to starvation.**