L-3/T-2/CSE **Date: 29/03/2023**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2020-2021

Sub: **CSE 313** (Operating System)

Full Marks: 210 Time: 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE** questions.

1. (a) Draw the Processes State Transition Diagram. Include the following states: *Embryo, Running, Runnable, Zombie, Sleeping.* **(10)**

(b) Given a basic spinlock, Assume that locking the spinlock takes **A** time units (if no one is holding the lock); unlock also takes **A** time units. Assume further that a context switch takes **C** time units, and that a time slice is **T** time units long.

Assume this code sequence, executed by **two** threads on **one** processor at roughly the same time: **(5×3=15)**

```
mutex_lock() ;
do_something() ;   // takes no time to execute
mutex_unlock() ;
```

i. What is the best-case time for the two threads on one CPU to finish this code sequence?

ii. What is the worst-cast time for the two threads to finish this code sequence? Assume that only **three** context switches can occur at a maximum.

iii. If the spin lock is instead changed to a queue-based lock, how does that change the worst-case time?

(c) You are given a new atomic function, called **`FetchAndSubtract()`.** It executes as a single atomic instruction, and is defined as follows: **(10)**

```
int FetchAndSubtract  (int *location )  {
    int value = *location;        // read the value pointed to by location
    *location = value - 1;        // decrement it and store result back
    return value ;                // return old value
}
```

You are given the task: write the **`lock_init()`, `lock()`,** and **`unlock()`** functions (and also Define a **`lock_t`** structure) that use **`FetchAndSubtract()`** to implement a working lock.

Contd …… P/2

CSE 313

2. (a) The variable **counter**, is shared within process A and process B. The initial value is `counter = 0` before execution of either process. Here, `R0` is a register. **(7+8=15)**

Process A	Process B
LOAD (counter, R0)	LOAD (counter, R0)
ADD (R0, 1, R0)	ADD (R0, 2, R0)
STORE (R0, counter)	STORE (R0, counter)

i. Add semaphores (with initial values) so that the final value of counter **is 2.**

ii. Add semaphores (with initial values) so that the final value of counter is **not 3.**

(b) You wrote a piece of code with four threads (1-4) and four locks (A-D).

Thread 1 grabs Locks A and B (in some order); Thread 2 grabs Locks B and C (in some order);

Thread 3 grabs Locks C and D (in some order); Thread 4 grabs Locks D and A (in some order);

Is it possible that this code might result in deadlock? Briefly explain. **(5)**

(c) Consider the following program: **(5+10=15)**

```
1    int main{
2        int count =1;
3        int pid = 0, pid2 = 0;
4        if ((pid = fork())) {
5            count = count + 2;
6            printf("%d ", count);
7        }
8
9        if (count == 1) {
10           count++;
12             pid2=fork();
13           printf("%d ", count);
14       }
15
16       if (pid2) {
17           wait(pid2, NULL, 0);
18           count = count * 2;
19           printf("%d ", count);
20       }
21   }
```

i. How many processes are created during the execution of this program? Explain briefly.

ii. List all the possible outputs of the program.

3. (a) Let's examine a program having two threads: **(5)**

Thread 1	Thread 2

pending = 1;
while (pending) {
printf("hello\n");
}
	pending = 0;

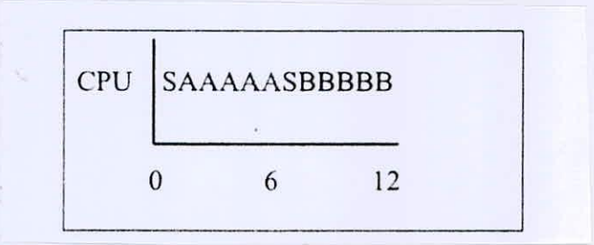
How could we re-write the code such that Thread 2 would only run after "hello" has been printed at least twice? You can use any synchronization primitive of your choice.

Contd P/3

CSE 313

Contd... Q. No. 3

(b) Scheduling policies can be easily depicted with some graphs. For example, let's say we run scheduler **S** for 1 time unit, job **A** for 5 time units, run scheduler **S** again for 1 time unit, and then run job **B** for time units. Our graph of this policy will look like this: **(5×3=15)**



i. Draw a similar graph of **ROUND-ROBIN** scheduling for jobs **A** (arriving at **T=0**), **B** (arriving at **T = 5**), and **C** (arriving at **T = 10**), each running for **6 time-units**. Assume a **2 time unit** time slice; also assume that the scheduler (**S**) takes 1 time unit to make a scheduling decision. Make sure to label the x-axis appropriately.

ii. What is the *average* ***RESPONSE TIME*** for jobs A, B and C?

iii. What is the *average* ***TURNAROUND TIME*** for jobs A, B and C?

(c) Consider the producer/consumer problem and the (broken) solution mentioned below. Briefly describe why solution is broken, and demonstrate it with a specific example of thread interleaving (*Hint: you can assume two consumers and one producer*). **(15)**

Producer	Consumer

`void *producer (void *arg) {`
`  int i;`
`  while (1) {`
`  mutex_lock (&mutex);  //p1`
`  if (count == MAX)  //p2`
`  cond_wait(&empty, &mutex);  //p3`
`  put (i);  //p4`
`  cond_signal (&full);  //p5`
`  mutex_unlock (&mutex);  //p6`
`  }`
`}`
	`void *consumer (void *arg) {`
	`  int i;`
	`  while (1) {`
	`  mutex_lock (&mutex);  //c1`
	`  if (count == 0)  //c2`
	`  cond_wait (&full, &mutex);  //c3`
	`  int tmp = get ();  //c4`
	`  cond_signal (&empty);  //c5`
	`  mutex_unlock (&mutex);  //c6`
	`  printf ("%d\n", tmp);`
	`  }`
	`}`

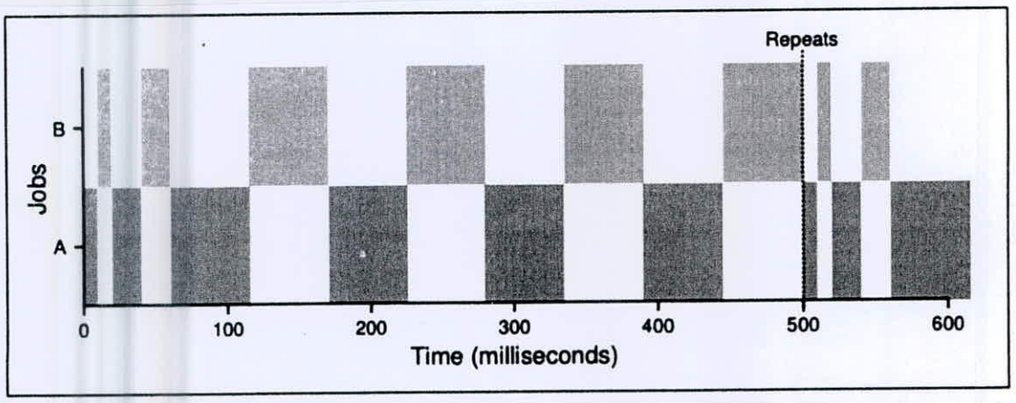
4. (a) A typical OS provides some APIs to create processes. **fork(), exec(),** and **wait()** can be used combinedly for that purpose. Write some code that uses these system calls to launch a new child process, have the child executed a program named "hello" (with no arguments), and have the parent wait for the child to complete. **(10)**

(b) Assume an OS with MLFQ (multi-level feedback queue) scheduler.

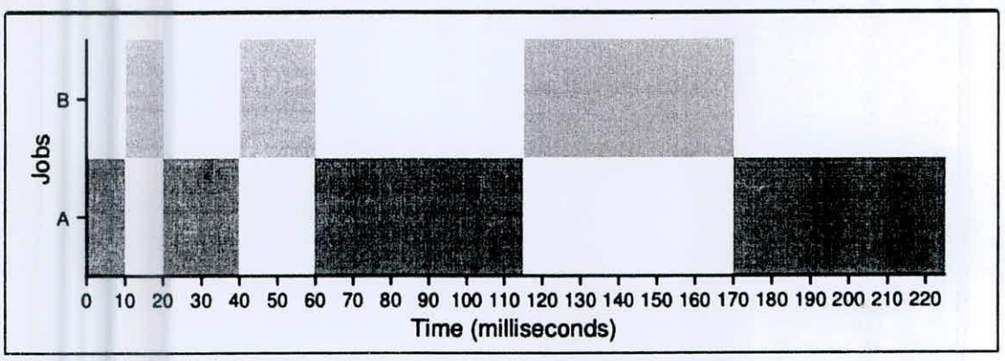
Here is a timeline of what happens when two CPU-bound (no I/O) jobs, **A** and **B** run: **(4×5=20)**

CSE 313

Contd... Q. No. 4(b)



This figure shows when A and B run over time. Note that after 500 milliseconds, the behavior repeats, indefinitely (until the jobs are done). To help you further, a closeup of the first part of the graph is shown below:



Now, answer the following questions:

i. How many queues do you think there are in this MLFQ scheduler?

ii. How long is the time slice at the top-most (high priority) queue?

iii. How long is the time slice at the bottom-most (low priority) queue?

iv. How often do processes get moved back to the topmost queue?

v. Why does the scheduling policy MLFQ move processes to higher priority levels (i.e., the topmost queue) sometimes? Briefly explain.

(c) With the round robin (RR) scheduling policy, a question arises when a new job arrives in the system: should we put the job at the front of the RR queue, or the back? Does this subtle difference make a difference, or does RR behave pretty much the same way either way? Briefly explain. **(5)**

Contd P/5

CSE 313

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE**.

5. (a) Alice has implemented a fast file system (FFS) with inodes having 12 direct pointers, 1 indirect pointer and 1 double indirect pointer. **(15)**

i. In a 1TB disk with 2KB blocks, how big of a file can be handled using Alice's file system?

ii. FFS divides the disk into multiple block groups. Now, can a 100 MB file be saved in Alice's file system? If so, then how all the data will be distributed on disk?

iii. If the disk spends 5ms time for positioning on the average and has 100MB/s transfer rate, how long will it take to read the file in (ii)?

(b) What is TLB in the context of memory virtualization? Show its structure. Mention its control flow using pseudocode. How does it manage context switches? **(10)**

(c) What is RPC? Suppose the user has written the following codes: **(10)**

```
int main() {
    ...
    m = f1(x1, x2);
    ...
}
int f1(char* a, char* b){
    ...
    p = f2(*a);
}
float f2(int t){
    ...
}
```

If the user wishes to run the function `f2` in a distributed manner, how will a RPC run-time library handle it? Write down which steps it needs to take and corresponding details that are needed to be taken care of in each step.

6. (a) Suppose the OS made the following 3 batch of block requests to a disk (each number represents block id): **(15)**

- 1, 40, 2, 15
- 10, 1, 13, 32, 2, 7
- 70, 49, 0, 6, 28

Contd P/6

CSE 313

Contd... Q. No. 6(a)

The requests were sent in such a way that one batch is requested after the previous one is handled. The disk has 1GB capacity, 4KB blocks, 8 blocks in each track and max seek time of 100ms. Now, calculate how much time the disk will spend on seeking, if the disk head is initially on the first track and the scheduling algorithm is

i. SSTF

ii. SCAN

iii. C-SCAN

(b) A program named `'test'` execute some code and writes some test to the console. The program is executed as: **(10)**

```
./test > /etc/out.txt
```

Now, write down the timeline for read/write operation in the file system. Assume that it is vsfs (very simple file system) and there is no cached value to aid.

(c) Bob has created the following files in an FFS (file sizes are written inside brackets). **(10)**

- `/a/f1 (2KB)`
- `/b/f2 (1KB)`
- `/c/b/f3 (10 KB)`
- `/c/f4 (7KB)`
- `/b/f5 (60KB)`

Suppose, the disk has 4KB blocks and 5 block group. If `/c/b` is symbolic link to the directory `/b`, show how the files will be saved with illustrative diagrams.

7. (a) You created an elegant device that has to deal with both big bursts of small I/O requests and very large I/O requests. While writing a device driver for it, what design decisions would you take and why? Elaborate the pros and cons of your design. **(15)**

(b) We need to write data in block 8, 11, 12 inode is in block 3, bitmap is in block 2. Journal is in blocks 24 to 31. Write down the journaling timeline if the system uses – **(10)**

i. Data Journaling

ii. Metadata Journaling

iii. Metadata Journaling with Checksum Optimization

(c) What is track skew? Why was it required in older disks? Why is it not good for newer disk models? **(5)**

(d) You know that RAID-4 has terrible Random Write performance. If the parity disk is mirrored in RAID-4, what would happen to its Random Write performance? What about Random Read or Sequential Read-Write? Explain your derivation. **(5)**

Contd P/7

CSE 313

8. (a) You ran the following rename operation: **(20)**

```
mv ~/a/foo.txt ~/b/bar.txt
```

Explain with illustrative figure what happens during this operation in-

i. Very Simple File System (VSFS)

ii. Log-structured File System (LFS)

(b) Using illustrative figures and pseudocodes, show how the OS talks to a canonical IO device while using DMA. **(10)**

(c) When using the swapping mechanism, the OS reserves some swap space on disk and works with directly. However, we know the OS already has File API to communicate with disk. Why do you think this discrepancy is there? **(5)**

L-3/T-2/CSE **Date: 18/04/2022**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2019-2020

Sub: **CSE 313** (Operating Systems)

Full Marks: 210 Time: 3 Hours

The figures in the margin indicate full marks

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE** questions.

1. (a) Write down the steps of booting a computer. **(8)**

(b) Consider the code written in C shown in the following figure. Here `fork()` is an UNIX system call that creates a child process identical to the parent. Executing this code will generate a process tree. Each of the created processes will have its own copy of variable *i*. Your task is to draw this process tree. At each node of the tree, you have to mention the starting value of *i* for the corresponding process. Consider that the root process is called P0. **(10)**

```
int i=0;
int main(){
        for(;i<3;i++){
                fork();
        }
        return 0;
}
```

(c) Write down the steps in making a system call. **(10)**

(d) What are the advantages of hybrid implementation of threads? **(7)**

2. (a) State the four conditions of resource deadlock. **(6)**

(b) State the problem definition of the classical Dining Philosophers problem. Show that the problem meets all the four conditions for resource deadlock stated in 2 (a). **(12)**

(c) A solution to the Dining Philosophers problem is given in Figure for Q2(c). **(12)**

(i) Suppose that in the `put-forks(i)` function, variable `state[i]` was set to `THINKING` **after** the two calls to test, rather than before. How would this change affect the solution? Explain with an example.

(ii) Suppose that in the `test(i)` function, **`up`**`(&s[i])` is placed outside the "if" condition. How would this change affect the solution?

Contd P/2

CSE 313

Contd... Q. No.2

(d) Explain the race-condition that exists in the following solution to the producer-consumer problem. **(5)**

```
#define N = 100
int count = 0;
void producer (void) {
   int item;
   while (TRUE){
      item = produce();
      if (count == N) sleep();
      inset(item);
      count = count + 1;
      if (count == 1)
          wakeup(consumer);
}}
```

```
void consumer (void) {
int item;
while (TRUE){
    if (count == 0) sleep();
    item = remove-item();
    count = count - 1;
    if (count == N-1)
       wakeup(producer);
    consume(item);
}}
```

3. (a) A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows: **(10)**

	Allocated	Maximum	Available
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

What is the smallest value of x for which this is a safe state? Show the intermediate steps.

(b) What is the difference between livelock and starvation? **(5)**

(c) Draw the resource graph for the following scenario where A, B, C, D, and E denote processes and 1, 2, 3, 4, and 5 denote resource types. There exists only one resource of each type. Show the steps of the execution of the deadlock detection algorithm on the constructed graph starting from node B. **(20)**

(i) Process A holds 1 and 3, wants 2
(ii) Process B holds 4, wants 3 and 5
(iii) Process C holds nothing, wants 2
(iv) Process D holds nothing, wants 2
(v) Process E holds 5 and wants 1

4. (a) Discuss the difference between compute-bound and I/O bound process with figures. **(6)**

(b) Consider the following workload: **(24)**

Process	Priority	Duration (sec)	Arrival Time (sec)
P1	3	70	40
P2	1	40	0
P3	1	100	10
P4	2	50	70

Contd P/3

CSE 313

Contd... Q. No.4(b)

Draw the Gantt chart and calculate the average turnaround time for each of the following scheduling algorithms:

(i) Non-preemptive Shortest Job First

(ii) Round Robin with quantum 30 sec. Consider the processes enter FIFO queue according to their approval times.

(iii) Priority Scheduling. Within the same priority class, schedule according to the scheduling algorithm mentioned in (ii).

(c) Write down the goals of scheduling algorithms for Batch Systems. **(5)**

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE** questions.

Unless otherwise specified, 1 k = 2^{10}, 1 M =2^{10}, 1 G =2^{30}.

5. (a) In 64-bit systems, 48 bit addressing is usually used. Calculate the amount of space needed (in GB) for a single-level page table for 48 bit addressing if the page size is 4 kB and each page table entry takes 4 bytes. Discuss the feasibility of such a page table and mention two (2) better alternatives. **(6+4=10)**

(b) Describe the advantages and disadvantages of memory-mapped I/O. **(10)**

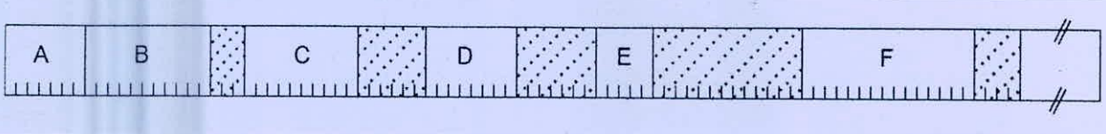
(c) Draw an omega switching network for 8 CPUs and 8 memory modules. Determine whether the following requests can be processed in parallel in this omega switching network and explain the reason. **(7+8=15)**

Request from CPU 000 to Memory 111 and

Request from CPU 110 to Memory 100

6. (a) The i-node of a Unix-like file system has 12 direct, one single-indirect and one double-indirect pointers. The disk block size is 4 kB and the disk block address is 32-bits long. Calculate the maximum possible file size for this file system. (Note that a single-indirect pointer points to a block of pointers that then point to blocks of the file's data. A double-indirect pointer points to a block of pointers that point to other blocks of the pointers that then point to blocks of the file's data.) **(10)**

(b) In the snapshot of memory given below, the dotted areas indicate holes and A, B, C, D, E, F are processes currently in memory. Create a linked list of processes and holes for swapping, assuming that there is no virtual memory. **(5+5=10)**



Contd P/4

CSE 313

Contd... Q. No.6(b)

A new process named G has to be allocated in memory now. The size of G is 4 allocation units. Mention the hole where G will be allocated if we follow:

(i) First-fit

(ii) Best-fit

(c) Define stable storage and mention the assumptions associated with it. Describe the three (3) basic operations a stable storage ensure. **(6+9=15)**

7. (a) Define precise interrupt and mention its four (4) properties. Describe the disadvantages of precise interrupts. **(5+5=10)**

(b) Mention the three (3) most essential properties of files. Explain how contiguous allocation of files may create both internal and external fragmentation in disk. **(3+7=10)**

(c) Mention the three key characteristics of a NUMA machine. In a directory-based NUMA multiprocessor having 1024 nodes, each memory address contains 48 bits. Each node consists of one CPU and 4 GB of RAM connected to the CPU via a local bus. Each cache line is 128 bytes long. The memory is statically allocated among the nodes. Calculate the directory overhead in percentage with respect to memory and discuss the feasibility of this system. **(3+12=15)**

8. (a) In a hypothetical machine, there are a total of 8 physical page frames. **(10)**

Page Table Index	Time of Last Use (t)	Referenced Bit	Modified Bit
0	214	1	1
1	381	0	1
2	402	1	0
3	289	1	1
4	409	0	0
5	160	1	1
6	315	1	0
7	387	0	1

At t = 430 and t = 440, two page faults occur. Using working set page replacement algorithm (with execution time approximation), identify the page frames that will be evicted. Assume the age threshold (τ) to be 50. Also assume that the clock interrupt to clear referenced bit last occurred at t = 425, and this interrupt occurs at 20 units time interval.

(b) Describe one (1) major advantage and one (1) major disadvantage of DMA. Briefly explain the major functions of device independent I/O software. **(6+14=20)**

(c) Write the full forms of the following abbreviations: **(5)**

NFTS, SMP, ECC, MBR, UDF

```
#define N           5                   /* number of philosophers */
#define LEFT        (i+N-1)%N           /* number of i's left neighbor */
#define RIGHT       (i+1)%N             /* number of i's right neighbor */
#define THINKING    0                   /* philosopher is thinking */
#define HUNGRY      1                   /* philosopher is trying to get forks */
#define EATING      2                   /* philosopher is eating */

typedef int semaphore;                  /* semaphores are a special kind of int */
int state[N];                           /* array to keep track of everyone's state */
semaphore mutex = 1;                    /* mutual exclusion for critical regions */
semaphore s[N];                         /* one semaphore per philosopher */

void philosopher(int i)                 /* i: philosopher number, from 0 to N-1 */
{
    while (TRUE) {                      /* repeat forever */
        think();                        /* philosopher is thinking */
        take_forks(i);                  /* acquire two forks or block */
        eat();                          /* yum-yum, spaghetti */
        put_forks(i);                   /* put both forks back on table */
    }
}

void take_forks(int i)                  /* i: philosopher number, from 0 to N-1 */
{
    down(&mutex);                       /* enter critical region */
    state[i] = HUNGRY;                  /* record fact that philosopher i is hungry */
    test(i);                            /* try to acquire 2 forks */
    up(&mutex);                         /* exit critical region */
    down(&s[i]);                        /* block if forks were not acquired */
}

void put_forks(i)                       /* i: philosopher number, from 0 to N-1 */
{
    down(&mutex);                       /* enter critical region */
    state[i] = THINKING;                /* philosopher has finished eating */
    test(LEFT);                         /* see if left neighbor can now eat */
    test(RIGHT);                        /* see if right neighbor can now eat */
    up(&mutex);                         /* exit critical region */
}

void test(i) /* i: philosopher number, from 0 to N-1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        up(&s[i]);
    }
}
```

Figure for Q 2(c)

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2018-2019

Sub: **CSE 313** (Operating Systems)

Full Marks: 180 Section Marks: 90 Time: 2 Hours (Sections A + B)

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE**.

1. (a) A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes each. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the machine's address space? Suppose that instead of 4096 bytes, the page size were 512 bytes, would it then fit? Each page must contain either text, data, or stack, not a mixture of two or three of them. **(10)**

(b) Write an argument for and an argument against small page size. Derive the formula for determining the optimal page size in a system in terms of average process size (s) and each page table entry (e). **(4+6)**

(c) A small computer on a smart card has four page frames. At the first clock tick, the R (reference) bits are 0111 (page 0 is 0, the rest are 1). At subsequent clock ticks, the values are 0010, 1010, 1100, 1011, 1010, 1101, and 0001. If the aging algorithm is used with an 8-bit counter, give the values of the four counters after the last tick. If a page has to be replaced now, which page will be selected. **(8+2)**

2. (a) Describe the architecture and write down the advantages and disadvantages of RAID level 2. **(10)**

(b) Suppose disk requests come in to the disk driver for cylinders 8, 30, 20, 19, 40, 16, and 50, in that order. A seek takes 6 msec per cylinder. What is the total seek time needed if the Elevator algorithm is used for disk arm scheduling? Given, the arm is initially at cylinder 20 and moving upward. **(10)**

(c) Explain how a system can simulate multiple virtual clocks with a single physical clock. Use the following example for your explanation. The current time is 3000 and there are pending clock requests for time 3006, 5014, 5019, 5025, and 5039. **(10)**

3. (a) Describe the four conditions that are necessary for a resource deadlock to occur. Give an example to show that these conditions are not sufficient for a resource deadlock to occur. **(6+4)**

(b) Write down the four strategies that are used for dealing with deadlocks. Describe some ways of recovering from deadlocks. **(4+6)**

(c) Consider the following snapshot of a system. There are three processes, P1, P2 and P3 competing for three resources R1, R2, and R3. **(8+2)**

Total Resource Instances		
R1	R2	R3
3	2	5

Available		
R1	R2	R3
1	1	1

Process ID	Allocated		
	R1	R2	R3
P1	0	0	2
P2	1	0	2
P3	1	1	0

Process ID	Maximum Required		
	R1	R2	R3
P1	3	1	3
P2	1	2	3
P3	2	2	1

Now, answer the following questions using the banker's algorithm:

(i) Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.

(ii) If a request from process P1 arrives for (1, 1, 1), can the request be granted immediately?

4. (a) What is the problem of using single level page tables for very large virtual address spaces? How can this problem be eliminated using multilevel page tables? Explain with a suitable example. **(10)**

(b) What is the difference between 'cycle stealing' and 'burst mode' in the context of DMA operation? Describe three techniques that an operating system can use to reduce power consumption in hard disks. **(4+6)**

(c) Why is Banker's algorithm for deadlock avoidance not useful in practice? Describe how the circular wait condition can be eliminated in a system to prevent deadlocks. **(5+5)**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examination (January 2020 Term)

Subject: **CSE 313** (Operating System)

Full Marks: 180 Section Marks: 90 Time: 2 Hours (Sections A + B)

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE**.

5. (a) What are the steps of booting a computer? You can assume a BIOS system (NOT a UEFI system). **(12)**

   (b) What are the three cases when a CPU can operate in kernel mode? Briefly describe all three cases. **(3x6=18)**

6. (a) What are the four scenarios when a process may get created? Briefly explain all of them in one/two lines. **(4x3=12)**

   (b) Draw the process state diagram. **(9)**

   (c) What are the differences between a user thread and a kernel thread? **(9)**

7. (a) Explain the IPC primitive known as a barrier. Give an example scenario where a barrier can be useful. **(12)**

   (b) Explain the guaranteed scheduling algorithm. In particular, explain the goal of the algorithm, the metric used to determine which process will run next, the pros and cons. **(4+8+3+3=18)**

8. (a) What is a journaling filesystem? Give an example. **(12)**

   (b) How does UNIX-V7 handle very large files? Explain with proper figures. **(18)**

L-3/T-2/CSE **Date : 31/03/2019**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2017-2018

Sub : **CSE 313** (Operating System)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE**.

1. (a) Briefly describe first fit and next fit memory management algorithms. Which one performs better in practice? Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of (i) 12 MB, (ii) 10 MB, and (iii) 9 MB for best fit and worst fit? **(10)**

   (b) Suppose that a machine has 38-bit virtual addresses and 32-bit physical addresses. Now answer the following in this context. **(10)**

   i. What is the main advantage of a multilevel page table over a single-level one?

   ii. With a two-level page table, 16-KB pages, and 4-byte entries, how many bits should be allocated for the top-level page table field and how many for the next level page table field?

   (c) What are the advantages of segmentation over paging? Describe with necessary diagrams how MULTICS incorporates segmentation with paging. **(10)**

   (d) The average process size is s bytes, the page size is p bytes, and each page entry is e bytes. Deduce the equation for optimal page size. **(5)**

2. (a) Suppose that the WSClock page replacement algorithm uses at τ of 2 ticks, and the system state is the following: **(10)**

Page#	Time stamp	V	R	M
0	9	1	1	0
1	9	1	1	1
2	7	1	0	0
3	4	0	0	0
4	6	1	0	1

   Here, the three flag bits V, R, and M stand for Valid, Referenced, and Modified, respectively.

   (i) If a clock interrupt occurs at tick 10, show the contents of the new table entries with explanation.

   (ii) Suppose that instead of a clock interrupt, a page fault occurs at tick 10 due to a read request to page 3. Show the contents of the new table entries with explanation.

Contd. P/2

CSE 313

Contd ... Q. No. 2

(b) A computer has four-page frames. The time of loading, time of last access, and the R and M bits for each page are as shown below (the times are in clock ticks): **(10)**

Page	Time of loading	Last ref.	R	M
0	230	265	0	1
1	140	270	0	0
2	110	285	1	1
3	126	280	1	0

i. Which page will NRU algorithm replace?

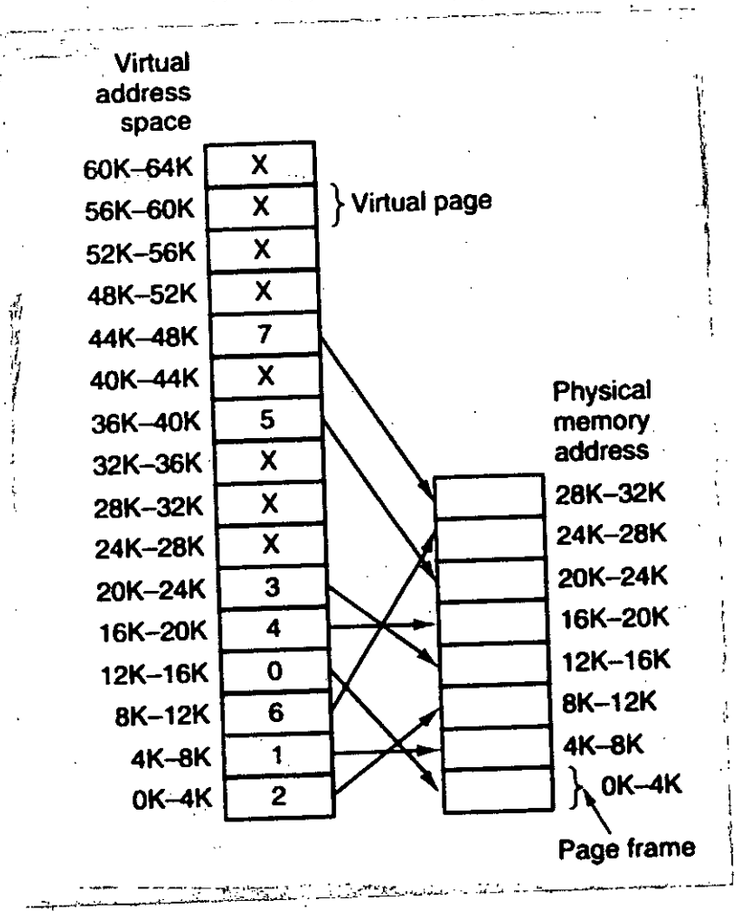
ii. Which page will FIFO algorithm replace?

iii. Which page will LRU algorithm replace?

iv. Which page will second chance algorithm replace?

(c) "Page fault cost varies" - is this statement true or false? Give appropriate reasoning of your answer. Briefly describe copy on write technique for the fork system call. **(10)**

(d) Using the page table below, give the physical address corresponding to each of the following virtual addresses: (i) 20, (ii) 4100, and (iii) 8300 **(5)**



CSE 313

3. (a) Briefly describe resource allocation graph with examples. Write down the algorithm that uses resource allocation graph to detect deadlock with one resource of each type. **(10)**

(b) Write down the algorithm for deadlock detection with multiple resources of each type with describing each of the matrices (C, R, E, A) the algorithm requires. **(10)**

Consider the following state of a system with four processes, P1, P2, P3, ad P4, and five types of resources, RS1, RS2, RS3,RS4, and RS5:

C =

0	1	1	1	2
0	1	0	1	0
0	0	0	0	1
2	1	0	0	0

R=

1	1	0	2	1
0	1	0	2	1
0	2	0	3	1
0	2	1	1	0

E=

2	4	1	4	4

A=

0	1	0	2	1

Using the above deadlock detection algorithm, find out whether there is a deadlock in the system and identify the processes that are deadlocked.

(c) Briefly describe the four conditions for resource deadlock. By attacking which condition, you can achieve the most feasible deadlock prevention technique? How can you prove the correctness of this technique? **(10)**

(d) Two processes, A and B, each need three records, 1, 2, and 3, in a database. If A asks for them in the order 1, 2, 3, and B asks for them in the same order, deadlock is not possible. However, if B asks for them in the order 3, 2, 1, then deadlock is possible. What fraction of all the combinations is guaranteed to be deadlock free? **(5)**

4. (a) Suppose you are writing a program to print your first name using the printer. Describe how you can achieve that using, (i) Programmed I/O, (ii) Interrupt driven I/O, and (iii) I/O using DMA with small code segments identifying key features. **(10)**

(b) Disk requests come in to the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. A seek takes 6 ms per cylinder. How much seek time is needed for (i) First-come, first served, (ii) Shortest Seek First, (iii) Elevator algorithm (initially moving upward). In all cases, the arm is initially at cylinder 20. **(10)**

(c) What is meant by stable storage? Describe three operations to accomplish stable storage. Prove with example that stable writes can survive CPU crashes. **(10)**

(d) What is a cache coherence protocol? What happens (based on this protocol) if a CPU attempts to write a word that is in one or more remote caches? **(5)**

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE**.

5. (a) What is **abstraction?** Briefly explain using an example how an operating system helps an application programmer by providing abstractions. **(5)**

(b) Briefly explain **dual mode operation.** How can a user program access services provided by an operating system? **(5+5=10)**

Contd. P/4

CSE 313

Contd ... Q. No. 5

(c) The following processes arrive at a system according to the table below. Assume that the process switching time is 0 second. Calculate **average turnaround** and **response time** for (i) FCFS Scheduling and (ii) Round Robin Scheduling with quantum of 2 seconds. **(10+10=20)**

Process	Arrival Time (sec)	CPU Burst Time (sec)
A	0	3
B	1.8	6
C	3.2	4
D	5.6	5
E	7.9	2

6. (a) Briefly describe the states of a typical process with a diagram. Mention how transitions occur between the states. **(10)**

   (b) Mention the advantages and disadvantages of implementing threads in the Kernel. **(4)**

   (c) How can you differentiate a program from a process? How a program becomes a process? **(6)**

   (d) What makes context switching a costly operation? How threads are helpful over processes in this regard? **(10)**

   (e) "Each user level thread needs its own stack" - do you agree with the statement? Why or why not? **(5)**

7. (a) How can you use semaphore to ensure (i) **mutual exclusion.** (ii) **access to limited resource,** and (iii) **synchronization?** Briefly explain each case with appropriate pseudo-code(s). **(15)**

   (b) When multiple co-operating processes communicate, special attention is required so that the system does not get stuck because of deadlocks. Providing proper synchronization mechanism to ensure liveliness and controlling access to shared resources are also some factors which makes the job quite difficult. **(20)**

   To pass this exam, you might have studied some of the classical IPC problems and their solution approaches. However here we are asking you to **design a new problem**. Think of a non-classical IPC problem. You don't need to solve the problem. Rather, **clearly mention the IPC issues in this problem.**

8. (a) What is a **race condition**? Why **cooperating processes** are necessary? **(5)**

   (b) What is **disk fragmentation**? Briefly explain how implementing files using contiguous allocation leads to this problem. **(10)**

   (c) Briefly describe a strategy to check consistency of blocks in a file system. **(10)**

   (d) Show the structure of an **i-node** in **UNIX V7** file system. **(10)**

L-3/T-2/CSE **Date : 12/03/2018**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2016-2017

Sub : **CSE 313** (Operating System)

Full Marks: 210 Time : 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE** questions.

1. (a) Draw the block diagram of the Unix System kernel and show the interaction among different subsystems. **(10)**

   (b) Why is it necessary to have two different stacks: user stack and kernel stack? Explain with an example. **(8)**

   (c) How is buffer list maintained in Unix system? Draw necessary diagram. What can happen to a process, if a buffer it is requesting is marked "delayed write"? **(7)**

   (d) Consider two events for which processes are at sleep states: (i) "Any buffer becomes free", (ii) "a specific buffer becomes free". If the specific buffer becomes free after some time, how is it handled by the operating system? Explain briefly. **(10)**

2. (a) Consider a case when a process P is waiting for a buffer #99 to become free (currently process Q is using it). Is it possible (due to some race condition) that suddenly process P discovers buffer #99 is not in the buffer cache list? Explain with necessary diagram. **(12)**

   (b) If somebody just changes the file access permissions (using chmod Command), what changes will be seen in file access times? Explain briefly. **(8)**

   (c) When a shortcut is made for a file, how many new inode(s) is used for this purpose? Give an example with respect to the changes in disk inode(s). **(7)**

   (d) "Superblock free inode list contains all the free inodes available in the disk" – do you agree with the statement? Why or why not? Explain. **(8)**

3. (a) Consider a modified table of contents (ToC) for inode having fourteen entries where the last entry is quadruple indirect pointer to data block, whereas other entries remain the same as before. If each data block is of 2K bytes and 32- bits are required for data block addressing, find out the maximum file size. **(10)**

   (b) Considering a similar structure as explained in Question # 3(a), locate the byte offset 4,50,000 in a file. Assume block numbers to explain and draw necessary diagram(s). **(8)**

   (c) What is the role of remembered inode while freeing an inode? Give examples. **(7)**

   (d) Write down the steps that need to be followed in wakeup system call. **(10)**

Contd P/2

CSE 313

4. (a) While writing to a file, is it possible that kernel allocates three new disk blocks (to just write one byte of data)? Explain the scenario with diagram. **(8)**

(b) Draw the process state transition diagram for Unix System. **(7)**

(c) What is the purpose of Register triple while managing regions? Show how it changes when kernel changes the size of a stack region by 2K bytes (consider each page to be 1K bytes). **(10)**

(d) Hoe does Kernel use u area in the Unix System? What are major information kept there? Explain briefly. If there are processes (P, Q, and R) running currently, how many u area are handled by kernel simultaneously? **(10)**

SECTION-B

There are **NINE** questions in this section. Answer any **SEVEN** questions.

5. (a) What is multiprogramming? Suppose you have a single CPU system. Will implementing multiprogramming improve the efficiency of your system? Justify your answer. **(6)**

(b) Compare the advantages and disadvantages of implementing threads in user space and kernel space. **(9)**

6. (a) What is meant by preemptive and non- preemptive scheduling algorithms? Give examples. **(4)**

(b) What is meant by **throughput** and **turnaround** time in process scheduling? Give examples. **(5)**

(c) Consider the following four processes: **(6)**

Process	Duration	Arrival time
P1	10	0
P2	4	5
P3	16	2
P4	6	4

Draw Gantt charts and then calculate average turnaround time and throughput for the processes using **First Come First Serve** scheduling algorithm.

7. (a) What is two phase locking? **(3)**

(b) Why multiple threads are used in a process instead of using multiple processes for the same task? **(5)**

(c) What is meant by the term **quantum** in context of process scheduling? What are the disadvantages if the quantum for scheduling is too large or too small? **(7)**

Contd P/3

CSE 313

8. (a) What is a multilevel page table? Explain with necessary figures. **(5)**

   (b) Describe a scenario when you would want to use multilevel page tables instead of single level page table. **(4)**

   (c) Do you think using multilevel page tables may reduce page faults in a system? Justify your answer. **(6)**

9. (a) Suppose you have four processes with length 48, 64, 120, and 28 bytes, respectively. The memory available for user space is 300 bytes. Discuss a way by which you can run all the four processes parallelly in your system minimizing number of memory accesses. **(5)**

   (b) What is meant by **protection** and **relocation**? Can both protection and relocation be ensured using Base and Limit registers only? Explain your answer. **(10)**

10. (a) Suppose you are going to design Memory Management Unit for a system having 8GB memory with 64-bit virtual address space. The system should be robust enough to allow any number of processes of arbitrary size to run parallelly. Now, briefly discuss the memory management technique you will use for ensuring protection and relocation in your system. Give arguments justifying your choice. **(8)**

    (b) Enumerate the four strategies used for dealing with deadlocks. Which one of them is the most popular and why? **(7)**

11. (a) What is meant by **deadlock** in operating system? Briefly discuss how deadlock can be prevented. **(15)**

12. (a) What is meant by safe and unsafe states in the context of process scheduling? **(3)**

    (b) Suppose a system consists of three different processes A, B, and C. There are in total 5 printers and 6 scanners available to the system which the processes can use in a non-preemptable way. Process A is currently using 2 scanners and needs 3 more printers and 3 scanners in order to complete. Process B is currently using 3 printers and 3 scanners and needs 2 more printers and 3 more scanners in order to complete. Finally, process C is currently using a printer and needs one more printer and a scanner in order to complete. From this information, determine whether the processes can be in deadlock or not. **(12)**

13. (a) What is meant by memory mapped I/O and port mapped I/O? Discuss the advantages and disadvantages of each of them. **(10)**

    (b) Write a short note on **Precise and Imprecise Interrupts**. **(5)**

L-3/T-2/CSE **Date : 16/08/2017**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2015-2016

Sub : **CSE 313** (Operating System)

Full Marks: 210 Time : 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE**.

1. (a) What are the main problems is a monolithic system? Explain with a necessary diagram. **(7)**

   (b) How does the file subsystem interact with hardware in Unix? Explain with respect to Unix System Kernel structure. **(10)**

   (c) Some buffers are marked as "delayed write". What does this mean and how are those handled while allocating a new buffer by Unix Kernel? **(10)**

   (d) What are the information kept in buffer headers? **(8)**

2. (a) Explain with necessary diagrams all possible scenarios while allocating a buffer (using getblk algorithm) when the buffer cache of free buffers is non-empty. **(12)**

   (b) What are the two types of inodes: disk inode and incore inode? What additional information are needed for incore inodes? Why are they absent in disk inode? Explain. **(8)**

   (c) What is the significance of inode link count while freeing an inode (in algorithm iput? What happens when inode reference count is zero but link count is non-zero? **(7)**

   (d) Write down the steps to convert filename into inode. **(8)**

3. (a) Consider that in a Unix system, each data block size is 4k bytes and 32 bits are required for data block addressing. Assuming usual table of contents structure of Unix System V, find out the maximum file size of such system. **(10)**

   (b) If 32-bytes are necessary to store information of a directly or a file, then how many files or subdirectories are possible inside a directly of a system explained in Question # 3(a). **(5)**

   (c) What is the basic difference of data structures for the following two: (i) Superblock free inode list (ii) Superblock free disk block lisk? Explain with necessary diagrams. Give the reason of having such different data structure. **(15)**

   (d) "Rememberd disk block number is used to search next available disk blocks" – Justify the statement. **(5)**

4. (a) What happens when a create file system call is invoked? Write down the algorithm and show the file data structures. **(10)**

Contd P/2

CSE 313

Contd... Q. No. 4

(b) Some region's size can be altered while some other's size cannot be altered (using growreg algorithm). Explain the reason and give examples. **(8)**

(c) Is it possible to have an interrupt within the service routine of another interrupt? Explain with respect to the context of a process. **(7)**

(d) What happens when the Kernel wakes up a sleeping process? Explain with necessary data structure. **(10)**

SECTION-B

There are **NINE** questions in this section. Answer any **SEVEN** questions.

5. (a) Show the states of a process and the possible transitions between the states in a diagram. Give an example for each possible transition. **(5)**

(b) Suppose you are going to write a program that will compute the sum of all possible 64 bit integers. Discuss how a multithreaded version of the program will affect the time required in comparison to that by single threaded program. **(6)**

(c) What do you mean by busy waiting? What are the problems of busy waiting? **(4)**

6. (a) What do you mean by CPU bound and I/O bound processes? Why I/O bound processes are given higher priority in priority scheduling algorithms? **(5)**

(b) Discuss the convey effect in First Come First Serve scheduling algorithm. **(4)**

(c) Consider the following code of producer process for a producer consumer system. **(6)**

```
semaphore empty = 10;
semaphore full = 0;
mutex m;
void producer(void) {
     int item;
     while(TRUE) {
          item = produce_item();
          lock(&m);
          down(&empty);
          insert_item(item);
          up(&full);
          unlock(&m);
     }
}
```

Will the above code work correctly? If not, explain the problems and write down the correct code for producer. If you think this code to be correct, write down the corresponding code for consumer.

Contd P/3

CSE 313

7. (a) Consider the following 4 processes: **(10)**

Process	Duration
P1	30
P2	10
P3	19
P4	56

Assume all processes arrived at the same time. Draw Gantt charts and then calculate average turnaround times for the processes using the following scheduling algorithms:

(i) Shortest Job First

(ii) Round Robin with quantum 8.

(b) Discuss the limitations of **Shortest Remaining Time Next** scheduling algorithm. **(5)**

8. Write short notes on the following terms: **(5×3=15)**

(i) Priority Scheduling

(ii) Guaranteed Scheduling

(iii) Lottery Scheduling.

9. (a) What do you mean by preempt able and non-preempt able resources? Briefly discuss the four conditions for deadlock. **(2+6=8)**

(b) Suppose a system is detected to be in deadlock. Discuss some ways to recover from it. **(7)**

10. (a) What are the problems of exposing physical memory to processes? **(4)**

(b) Suppose you are going to use bitmaps to keep track of memory usage. Discuss the issues you need to consider while determining the size of each allocation unit. **(5)**

(c) The **Not Frequently Used** (NFU) page replacement algorithm keeps a software counter associated with each page and increments it if the page had been referenced after previous clock interrupt. When a page fault occurs, the page with the lowest counter is chosen for replacement. Discuss a case where this algorithm will not perform optimally. Can you suggest a modification to the NFU algorithm to overcome this limitation? **(6)**

11. (a) Describe how deadlock can be detected by monitoring CPU utilization. **(3)**

(b) Suppose there are four processes P1, P2, P3, P4 and four resources R1, R2, R3, R4 in a system. The current allocation matrix and request matrix are as followed: **(12)**

Contd P/4

CSE 313

Contd... Q. No. 11 (b)

$$C = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 4 & 1 & 2 & 2 \\ 1 & 2 & 0 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 4 & 2 & 1 \\ 0 & 2 & 3 & 3 \\ 1 & 1 & 1 & 2 \\ 3 & 1 & 1 & 1 \end{bmatrix}$$

The availability resource vector is [3 2 2 1]. Determine whether the processes will be in deadlock or not.

12. (a) Suppose you have four processes with length 48, 64, 120, and 28 bytes respectively. The memory available to your system is 100 bytes. Discuss a way by which you can run all the four processes parallelly in your system. **(9)**

(b) What are the disadvantages if the page size is too large or too small? **(6)**

13. (a) What do you mean by **pre-paging** and **demand paging**? **(3)**

(b) What is TLB? Briefly describe how the TLB functions. **(7)**

(c) Briefly discuss how page sharing between processes can be handled by a memory management unit. **(5)**

L-3/T-2/CSE **Date : 27/07/2016**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2014-2015

Sub : **CSE 313** (Operating Systems)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE** questions.

1. (a) Operating systems need to play roles: **(15)**

(i) Referee

(ii) Illusionist

(iii) Glue

Identify the roles of operating systems for the following three tasks.

(i) Operating systems set up boundaries to "prevent bugs and malicious users and applications from affecting other users and their applications". However, "operating system must also allow those boundaries to be crossed in carefully controlled ways" so that various programs can effectively communicate with one another.

(ii) "Most operating systems also provide a standard way for applications to pass messages and to share memory."

(iii) Operating systems maintain virtual memory "to compensate for shortages of physical memory by temporarily transferring pages of data from random access memory (RAM) to disk storage."

(b) Suppose a computer system and all of its applications were completely bug free. Suppose further that everyone in the world were completely honest and trustworthy. In other words, we need not consider fault isolation. **(10)**

(i) How should an operating system allocate time on the processor? Should it give the entire processor to each application until it no longer needs it?

(ii) How should the operating system allocate its disk space? Should the first user to ask acquire all of the free spaces?

(c) "A central role of operating is protection – the isolation of potentially misbehaving applications and users so that they do not corrupt other applications or the operating system itself." Protection is essential to achieve the following goals of operating systems: **(10)**

(i) Reliability

(ii) Security

(iii) Privacy

(iv) Fairness

Contd P/2

CSE 313

Contd ... Q. No. 1(c)

Identify the goals of operating systems for the following two tasks:

(i) "An operating system must bullet proof itself to operate correctly regardless of what an application or user might do."

(ii) A smartphone operating system must ensure a third-party app could not access user's contact list without the user's consent.

2. (a) One of the hardware supports that operating system kernel needs is **Privileged Instructions** to protect applications and users from one another. Which of the following three instructions do you think are privileged? **(10)**

(i) "Adding two registers together and storing the result in a third register"

(ii) "Setting up a page table for a process"

(iii) "Put CPU in idle state using halt"

(b) Operating systems implement "dual-mode operation, represented by a single bit in the processor status register that signifies the current mode of the processor. In user mode, the processor checks each instruction before executing it to verify that it is permitted to be performed by that process. In kernel mode, the operating system executes with protection checks turned off." However, there is also another need of "safely transition from executing a user process to execute the kernel, and vice versa." These two transitions are known as "User to Kernel Mode Transfer" and "Kernel to User Mode Transfer" respectively. A special type of kernel to user mode transfer is "Upcall". From the following scenarios you need to identify the examples of user to kernel mode transfer, kernel to user mode transfer, and upcalls. **(16)**

i. **Resume after an interrupt**: When handling the request is finished, "the execution of the interrupted process is resumed by restoring its program counter."

ii. **Processor Exception**: "A processor exception is a hardware event caused by user program behavior after which the hardware finishes all previous instructions, saves the current execution state, and starts running at a specially designated exception handler."

iii. **New process**: To start a new process, its program is copied into memory, the program counter is set to the first instruction, the stack pointer is set to the top of the stack, and then the process is started.

iv. **Resource allocation**: "Operating systems allocate resources - deciding which users and processes should get how much CPU time, how much memory, and so forth. In turn, many applications are resource adaptive – able to optimize their behavior to differing amounts of CPU time or memory. An example is Java garbage collection.

Contd P/3

CSE 313

Contd ... Q. No. 2(b)

Within limits, a Java process can adapt to different amounts of available memory by changing the frequency with which it runs its garbage collector.

The more memory, the less time Java needs to run its collector, speeding execution. For this, the operating system must inform the process when its allocation changes, e. g., because some other processes need more or less memory."

(c) What is saved/restored on a process context switch if memory is managed using- **(9)**

(i) virtually addressed base and bounds

(ii) segmentation

(iii) paging

3. (a) Why do you think process management approach implemented in Windows is "simple in theory, however, complex in practice"? Explain. **(10)**

(b) Consider the following program: **(10)**

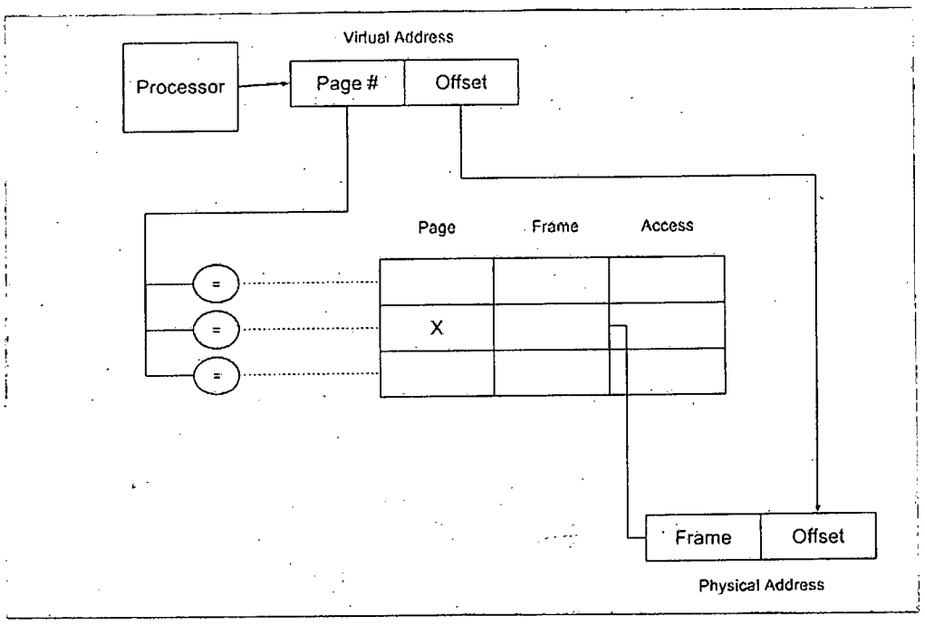
```
int main(int argc, char *argv[]){
    int child = fork();
    int x = 10;
    if(child){
        x += 10;
    } else {
        child = fork();
        x += 10;
        if(child == 0) {
            x += 10;
        }
    }
    return 0;
}
```

How many different copies of the variable **x** are there? What are their values when their processes finish?

(c) Illustrate virtually and physically addressed caches in a single block diagram. Suppose you have only one cache available. What addressing scheme would you use in this cache, virtually or physically addressed? Justify your answer. **(15)**

CSE 313

4. (a) Update the following translation lookaside buffer so that it can support multiple processes along with context switch. **(10)**



(b) Consider the following cache usage scenarios. Which replacement policies should we use for each usage scenario? **(8)**

	Random	FIFO	LRU	LFU	MRU
Web server cache					
Browser DNS cache					
Email client inbox					
Looping memory references					

(c) Illustrate the Cache Write Concept with block diagram. **(7)**

(d) For each of the following statement, indicate whether the statement is true or false, and also explain why. **(10)**

(i) "A virtual memory system that uses paging is vulnerable to internal fragmentation."

(ii) "A direct mapped cache can sometimes have a higher hit rate than a fully associative cache with FIFO replacement policy."

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE** questions.

5. (a) Describe the four stages of growth an NTFS file can go through with appropriate figures. **(15)**

(b) Discuss the characteristics of FFS multi-level index structure. **(10)**

Contd P/5

CSE 313

Contd ... Q. No. 5

(c) Draw the software layers that provide file system abstraction. What is the role of device drivers? **(5+5=10)**

6. (a) State and compare the locality heuristics of FAT, FFS, and NTFS file systems. **(15)**

   (b) Compare the efficiency of random access in FAT and FFS file systems. **(10)**

   (c) Provide an example where application libraries wrap system calls to add an additional functionality. **(10)**

7. (a) Consider the program on the left side and one possible output on the right side of the following figure. **(6×3=18)**

```
#include <stdio.h>
#include "thread.h"

static void go(int n);

#define NTHREADS 10
static thread_t threads[NTHREADS];

int main(int argc, char **argv) {
    int i;
    long exitValue;

    for (i = 0; i < NTHREADS; i++){
        thread_create(&(threads[i]), &go, i);
    }
    for (i = 0; i < NTHREADS; i++){
        exitValue = thread_join(threads[i]);
        printf("Thread %d returned with %ld\n",
                        i, exitValue);
    }
    printf("Main thread done.\n");
    return 0;
}

void go(int n) {
    printf("Hello from thread %d\n", n);
    thread_exit(100 + n);
    // Not reached
}
```

```
% ./threadHello
Hello from thread 0
Hello from thread 1
Thread 0 returned 100
Hello from thread 3
Hello from thread 4
Thread 1 returned 101
Hello from thread 5
Hello from thread 2
Hello from thread 6
Hello from thread 8
Hello from thread 7
Hello from thread 9
Thread 2 returned 102
Thread 3 returned 103
Thread 4 returned 104
Thread 5 returned 105
Thread 6 returned 106
Thread 7 returned 107
Thread 8 returned 108
Thread 9 returned 109
Main thread done.
```

Figure for Question 7(a)

Now answer the following questions.

(i) Why has the "Hello" message from thread 3 been printed before the "Hello" message from thread 2? Is it guaranteed that this sequence is always maintained?

(ii) Why has the "Thread returned" message from thread 3 been printed after the "Thread returned" message from thread 2? Is it guaranteed that this sequence is always maintained?

(iii) Why have the "Hello" message been merged with "Thread returned" messages?

(iv) What is the minimum and maximum number of threads that could exist when main thread prints "Thread returned" message?

(v) What is the minimum and maximum number of times that the thread 2 enters the READY state on a uniprocessor?

(vi) When **thread _join** returns for thread 3, what is the state of the main thread?

Contd P/6

CSE 313

Contd ... Q. No. 5

(b) What are the problems with a buggy thread? **(5)**

(c) Briefly explain Safety and Liveness in the context of "Too much Milk" problem. **(6)**

(d) "P( ) and V( ) functions of semaphore are commutative but wait( ) and signal( ) functions of condition variable are not." – Briefly explain. **(6)**

8. (a) For implementing lock in multiprocessor, why are disabling interrupts not enough? What extra measure should be taken? **(4+2=6)**

(b) Show that the Dining Philosophers problem meets all the four conditions for deadlock. **(8)**

(c) Briefly describe the scenario where both First-In-Out (FIFO) and Shortest Job First (SJF) scheduling work better than Round Robin scheduling. Why is space sharing more efficient then time sharing gin case of scheduling parallel applications? **(7+3=10)**

(d) What is the "Cache Coherence Overhead" problem in scheduling sequential applications on multiprocessors? How can this problem be solved? **(7+4=11)**

L-3/T-2/CSE **Date : 27/01/2016**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2013-2014

Sub : **CSE 313** (Operating System)

Full Marks: 210 Time : 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE**.

1. (a) Suppose the execution of a particular process requires reference to five distinct pages. The page reference sequence of the program is: **(15)**

2, 3, 2, 1, 5, 2, 4, 5, 3, 2 5, 2

which means that the first page referenced is 2, the second page referenced is 3, and so on. Assume that the physical memory contains three page frames and all the page frames are initially empty. Depict the behavior of each of the following page replacement algorithms for this process.

(i) Optimal page replacement algorithm

(ii) First-In First-Out (FIFO) page replacement algorithm

(iii) Least Recently Used (LRU) page replacement algorithm

You must show the state of physical memory after each page reference and calculate total page hit for each algorithm.

(b) Consider a computer system with 64 bits virtual address space. **(10)**

(i) Calculate the memory required to store a simple single-level page table for a single process given that

- Page size = 4 KB
- Physical memory size = 4 GB
- Each page table entry stores 12 bits of information in addition to 52 bits physical frame number

(ii) Calculate the memory required to store a single global inverted page table for all process given than

- Page size = 4 KB
- Physical memory size = 4 GB
- Process ID consists of 16 bits
- Each page table entry stores 12 bits of information in addition to virtual page number and process ID

(c) Suppose you are using paging technique for implementing virtual memory with 32 bit address space and 4 KB page size. You have decided to keep the entire page table for each process in main memory. As a result, you have to allocate a huge amount of memory for keeping the page table. Explain with necessary figures how you can solve this problem. You must prove the space efficiency of your technique. **(10)**

Contd P/2

CSE 313

2. (a) Suppose two processes P1 and P2 are running concurrently in a single processor system. The system contains two dedicated resources: one printer and one DVD-ROM. These resources can be used by only one process at a time. Both P1 and P2 has 600 instructions. P1 requests the DVD-ROM at 200th instruction and the printer at 400th instruction. P1 releases the DVD-ROM and printer at 300th and 500th instruction respectively. P2 requests the printer at 200th instruction and the DVD-ROM at 300th instruction. P2 releases the printer and DVD-ROM at 400th and 500th instruction respectively. **(10)**

Draw a diagram where each point depicts a joint state of the two processes where the horizontal axis represents the number of instructions executed by P1 and the vertical axis represents the number of instructions executed by P2. You must mark the regions where entry is restricted by the mutual exclusion rule.

(b) The following figure shows the state of a system consisting of four processes and three resources. The total amount of resources R1, R2, and R3 are 9, 3, and 6 units, respectively. Here, **R** is the total existing resource vector, **V** is available resource vector, and **A** is the current allocation matrix. The Claim matrix **C** shows the maximum requirement of each process for each resource. **(10)**

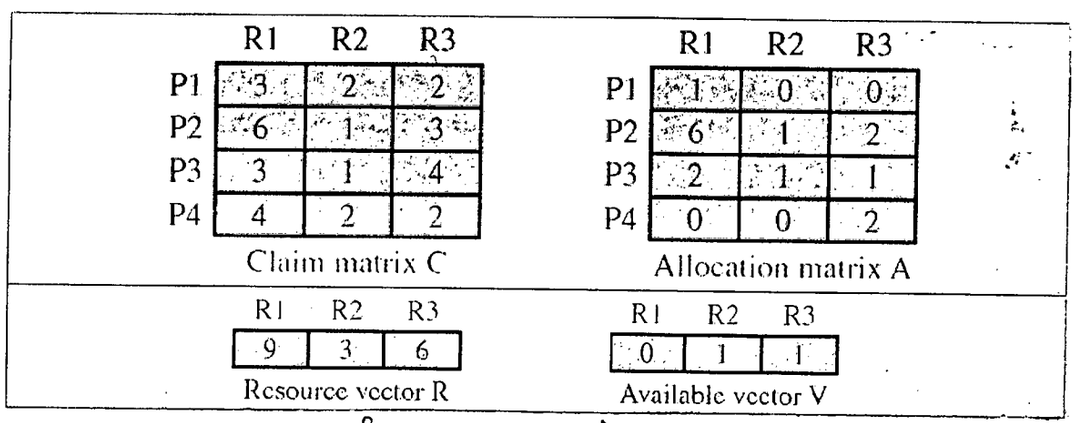


Figure for Q. No. 2(b)

(i) "The above system is in safe state"– Do you agree with this statement?

(ii) Provide proof for your answer?

(c) A particular CPU scheduling algorithm is implemented as follows: **(15)**

The ready queue of processes is treated as a FIFO queue. New processes are added to the tail of the ready queue as soon as they arrive. The scheduler removes a process from the head of the ready queue, sets a timer to interrupt after Q seconds, and gives control of the CPU to the process for execution.

One of two things will then happen:

(i) The currently running process may have a CPU burst of less than Q. In this case, the process itself will release the CPU voluntarily.

Contd P/3

CSE 313

Contd... Q. No. 2(c)

(ii) If the CPU burst of the currently running process is longer than Q, the timer will go off and will cause an interrupt to the operating system. The process will be added at the tail of the ready queue.

The scheduler will then remove another process from the head of the ready queue and proceed in the similar fashion.

Apply the above described algorithm for scheduling the workload given in the following table and illustrate the resulting schedule using Gantt chart for each of the cases:

(i) Q = 1

(ii) Q = 2

(iii) Q = 4

Assume that the process switching time is 0 second.

Process	Arrival Time (sec)	CPU Burst Time (sec)
A	0	3
B	1.8	6
C	3.2	4
D	5.6	5
E	7.9	2

Table for Q. No. 2(c)

3. (a) Consider the code written in C shown in the following figure. Here **fork()** is an UNIX system call that creates a child process identical to the parent. Executing this code will generate a process tree. Each of the created process will have its own copy of variable **i.** Your task is to draw this process tree. At each node of the tree you have to mention the starting value of **i** for the corresponding process. The root node is shown in the next figure. Draw the complete process tree appropriately. **(10)**

```
#include <stdio.h>
#include <unistd.h>
int i = 0;
int main()
{
    for (; i < 3 ; i++)
        fork();
    return 0;
}
```

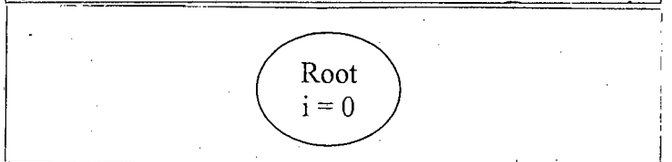


Figure for Q. No. 3 (a)

CSE 313

Contd... Q. No. 3

(b) The process state transition diagram for a uniprocessor system is shown in the following figure. **(15)**

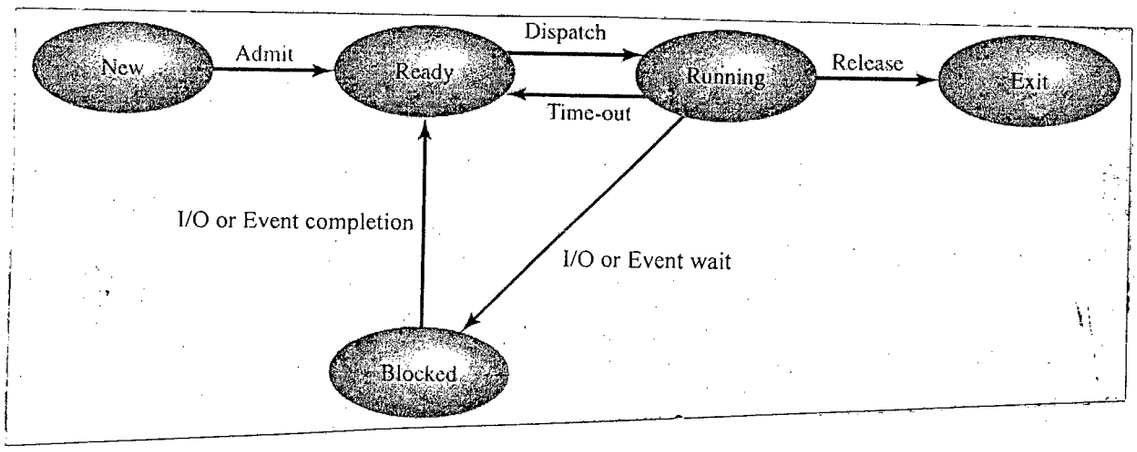


Figure for Q. No. 3(b)

Assume that there are only 5 processes named P1, P2, P3, P4 and P5 in the system. At time 0, each of the processes is in **Ready** state. The scheduler operates in round-robin fashion.

All events except dispatch that occur from time 5 to time 48 listed below:

- At time 5: P1 executes a command to read from disk.
- At time 15: P3's time slice expires.
- At time 18: P4 executes a command to write to disk.
- At time 20: P2 executes a command to read from disk.
- At time 23: P5 goes to sleep for 7 unit of time.
- At time 24: P3 executes a command to write to disk.
- At time 30: P5 wakes up from sleep.
- At time 33: An interrupt occurs from disk unit 2: P2's read is complete.
- At time 36: An interrupt occurs from disk unit 3: P1's read is complete.
- At time 38: P5 terminates.
- At time 40: An interrupt occurs from disk: P3's write is complete.
- At time 48: An interrupt occurs from disk: P4's write is complete.

Now infer the state of each process using the available information for each of the cases:

(i) At time 22

(ii) At time 37

(iii) At time 47

(c) For each of the page replacement algorithms listed below, mention the tasks which should be performed on each clock interrupt: **(10)**

CSE 313

Contd... Q. No. 3(c)

(i) First-In First-Out (FIFO): Replaces the oldest page

(ii) Second Chance: A simple modification to FIFO

(iii) Not Frequency Used (NFU): A crude approximation to LRU

(iv) Aging: A much better approximation to LRU

(v) WSClock: Based on working set information.

4. (a) Assume that a finite number of resources of a single resource type must be managed. Processes may ask for a number of these resources and will return them once finished. As an example, many commercial software packages provide a given number of licenses, indicating the number of applications that may run concurrently. When the application is started, the license count is decremented. When the application is terminated, the license count is incremented. If all licenses are in use, requests to start the application are denied. Such requests will only be granted when an existing license holder terminates the application and a license is returned. **(15)**

The following program segment is used to manage a finite number of instances of an available resource. The maximum number of resources and the number of available resources are declared as follows:

```
1.  #define MAX_RESOURCES 5
2.  int available_resources = MAX_RESOURCES;
```

When a process wishes to obtain a number of resources, it invokes the **`decrease_count()`** function:

```
3.  /* decrease available_resources by count resources */
4.  /* return 0 if sufficient resources available, */
5.  /* otherwise return -1 */
6.  int decrease_count(int count) {
7.        if (available_resources < count)
8.              return -1;
9.        else {
10.             available_resources -= count;
11.             return 0;
12.       }
13. }
```

When a process wants to return a number of resources, it calls the **`increase_count()`** function:

```
14. /* increase available_resources by count */
15. void increase_count(int count) {
16.       available_resources += count;
17. }
```

CSE 313

Contd... Q. No. 4(a)

The preceding program segment produces race condition.

(i) Identify the data involved in the race condition.

(ii) Mention the line number (or numbers) of the statement (or statements) in the above code which is (or are) responsible for the race condition.

(iii) Using a single binary semaphore, fix the race condition. You must rewrite the complete code. Make sure that your fixed version of **decrease_count()** returns 0 if sufficient resources are available and -1 otherwise.

(b) Te **decrease_count()** function in the question 4(a) currently returns 0 if sufficient resources are available and -1 otherwise. This leads to busy waiting for a process that wishes to obtain a number of resources: **(10)**

```
while (decrease count(count)== -1);
```

Implement a monitor named **resource_manager** which contains necessary data variables, condition variables and two functions named **void decrease_count(int count), void increase_count(int count)** such that **decrease_count()** function suspends the calling process until sufficient resources are available. This will allow a process to invoke **decrease_count()** by simply calling.

```
Resource_manager.decrease_count(count);
```

The calling process will return from this function call only when sufficient resources are available. You must maximize the degree of parallelism.

(c) Any facility or capability that is to provide support for mutual exclusion should meet the following requirements: **(10)**

(i) No two processes simultaneously in critical region

(ii) No assumptions made about speeds or numbers of CPUs

(iii) No process running outside its critical region may block another process

(iv) No process must wait forever to enter its critical region

A software solution to the mutual exclusion problem for two processes (Process 0 and Process 1) is proposed in Figure for Q. No. 4(c). Here the program fragments are written in C. The integer variable **truth**, initially 0, is shared by Process 0 and Process 1.

Process 0	Process 1

`while (TRUE) {`
`  while (turn != 0);`
`  critical_region( );`
`  turn = 0;`
`  noncritical_region( );`
`}`
	`while (TRUE) {`
	`  while (turn != 1);`
	`  critical_region( );`
	`  turn = 1;`
	`  noncritical_region( );`
	`}`

Figure for Q. No. 4(c)

Mention the conditions violated by this solution with proper explanation.

Contd P/7

CSE 313

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE**.

5. (a) Give a brief description of the followings: **(2½+2½=5)**

   (i) GNU/Linux OS

   (ii) Linux Distributions

   (b) Describe a scenario when two entries of a user file descriptor table may point to the same entry of the file table in UNIX. **(3)**

   (c) "While allocating an in-core inode in UNIX, the kernel either returns a locked in-core inode or returns an error" — Justify this statement using illustrative examples. **(6)**

   (d) Give a comparison of the free inode list and the free block list of the super block of a file system in UNIX. **(4)**

   (e) Answer the following question with respect to UNIX System V. **(2+3+2+4=11)**

   (i) Describe the inode table of content structure.

   (ii) If the logical block size of the file system is 1024 bytes and 4-byte integer is needed for data block addressing, what is the maximum byte capacity of a file?

   (iii) If the file size field in the inode is 8 bytes, what is the maximum file size of the operating system?

   (iv) Where can you find the 360,000 byte offset of a file in the system?

   (f) Describe the if-else statements in the given algorithm for releasing a buffer in UNIX using illustrative examples. **(6)**

```
algorithm brelse
input:  locked buffer
output: none
{
        wakeup all procs: event, waiting for any buffer to become free;
        wakeup all procs: event, waiting for this buffer to become free;
        raise processor execution level to block interrupts;
        if (buffer contents valid and buffer not old)
                enqueue buffer at end of free list
        else
                enqueue buffer at beginning of free list
        lower processor execution level to allow interrupts;
        unlock(buffer);
}
```

Algorithm for Question 5 (f)

Contd P/8

CSE 313

6. (a) "Linux is an example of UNIX operating system" — Justify this statement showing necessary comparisons. **(7)**

(b) Why the logged operations in journaling file systems must be idempotent? Explain with an illustrative example. **(5½)**

(c) Describe in-line and in-a-heap directory structure stating the advantages and disadvantages of each method using illustrative examples. **(4+4=8)**

(d) What happens to the different regions of a process when the **fork** system call is invoked by a process? Explain with respect to related UNIX kernel data structure. Use the illustrative figures are recommended. **(7)**

(e) Give a brief description of the followings: **(2½×3=7½)**

(i) Zombie state of a process in UNIX

(ii) Dynamic part of the system-level context of a process in UNIX

(iii) Memory-mapped I/O

7. (a) Assume, Shahid has a 10 MB solid-state drive in his PC. There is currently only one partition and no file in the SSD. The advertised block size of the SSD is 1 MB (1024 KB). Internally the file system layout of the SSD is as follows: **(3×3=9)**

0	1	2	3...9
MBR	Boot block	Super block / FATs	Data blocks

Shahid now wants to create 4 files as described in the following table in the only partition of the SSD.

File Name	Size
A	3.6 MB
B	1.0 MB
C	0.9 MB
D	0.6 MB

How will the file system (installed on his SSD) allocate hard disk blocks to the above files if it uses any of the following allocation schemes? Describe using illustrative figures.

(i) Contiguous allocation

(ii) Linked-List allocation

(iii) Linked-List allocation using a FAT in memory

Contd P/9

CSE 313

Contd... Q. No. 7

(b) After creating previous files, Shahid now wants to remove file A, C and create a file E of size 5 MB. How the file system installed on his SSD will behave to do so if it uses any of the following allocation schemes? Describe using illustrative figures. **(2×3=6)**

(i) Contiguous allocation

(ii) Linked-List allocation

(iii) Linked-List allocation using a FAT in memory.

(c) Describe the advantages and disadvantages of free space management using bitmap with illustrative examples. **(5)**

(d) Briefly describe the register context of a process in UNIX. **(3)**

(e) "Each buffer in UNIX always exist on a hash queue but not always on the free list" — Justify this statement. Why UNIX kernel maintains these separate data structures? **(3+3=6)**

(f) Why kernel stack needed in addition to user stack? Explain with an illustrative example. **(6)**

8. (a) Give a brief description of the followings: **(3×4=12)**

(i) Y2018 problem in MS-DOS file system

(ii) User-level context of a process in UNIX

(iii) Device Driver

(iv) Device Controller

(b) Describe the "continue" statements in the given algorithm for buffer allocation in UNIX using illustrative examples. **(9)**

```
algorithm getblk
input:  file system number
        block number
output: locked buffer that can now be used for block
{
    while (buffer not found)
    {
        if (block in hash queue)
        {
            if (buffer busy)
            {
                sleep (event buffer becomes free);
                continue;
            }
            mark buffer busy;
            remove buffer from free list;
            return buffer;
        }
```

CSE 313

Contd... Q. No. 8(b)

```
            else
            {
                if (there are no buffers on free list)
                {
                    sleep (event any buffer becomes free);
                    continue;
                }
                remove buffer from free list;
                if (buffer marked for delayed write) {
                    asynchronous write buffer to disk;
                    continue;
                }

                remove buffer from old hash queue;
                put buffer onto new hash queue;
                return buffer;
            }
        }
}
```

Algorithm for Question 8 (b)

(c) Describe the advantages and disadvantages of free space management using linked list with illustrative examples. **(5)**

(d) How does the UNIX kernel protect its consistency in kernel mode? Describe using illustrative examples. **(6)**

(e) Describe the differences between an in-core inode and a buffer header in UNIX. **(3)**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B.Sc. Engineering Examinations 2012-2013

Sub : CSE 313 (Operating System)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE**.

1. (a) Draw the block diagram of Unix system kernel. Explain how file subsystem interact with the hardware in Unix system. **(10)**

   (b) What are the purposes of using boot block, super block, i node list and data blocks in the Unix system? **(10)**

   (c) Why is kernel stack needed in addition to user stack? Explain using an example. **(8)**

   (d) What information are kept in buffer headers in Unix? **(7)**

2. (a) What is the main reason of keeping different (additional) information in incore inode compound to disk inode? Give examples of those additional fields. **(8)**

   (b) If a file is opened by two processes at the same time, how many incore indoes will be allocated? Explain with necessary diagram. **(7)**

   (c) Write the algorithm for allocation of incore inode. **(10)**

   (d) How can you find out the block number and the byte offset of an inode? If the starting block number of the inode list is 5, number of inodes per block is 32, and size of each disk inode is 72 bytes, then find out the block number and byte offset of inode number 555. **(10)**

3. (a) "While allocating buffer in Unix system, the kernel either returns a buffer after successful allocation or returns an error" - Justify this statement. **(6)**

   (b) Explain the case when **(5+5)**

   (i) The free disk block list superblock contains a single entry and some process requests for a new disk block.

   (ii) The free disk block list in superblock is full and a process frees another disk block.

   (c) What happens to the different regions of a process when fork system Call is invoked? Explain with respect to related Unix kernel data Structure. **(7)**

   (d) Consider a variant of Table of Contents Structure of Unix system Inode which have nine direct, one single indirect, one double indirect, one triple indirect and one quadruple indirect entry. Each data block size is 4K bytes and 64-bits are needed for data block addressing. What will be the maximum file size of such system? Where can you find the 1000,000 the byte offset of a file in such a system? **(12)**

Contd P/2

যে ব্যক্তি দাবী করে যে, সে এই দুনিয়া ও তার স্রষ্টাকে একই সাথে ভালবাসে সে আসলে মিথ্যা কথা বলে।
– ইমাম শাফি (র)

CSE 313

4. (a) Explain the algorithm for sleep system call. **(10)**

(b) List the major components of a process context. Use necessary figure to explain briefly. **(10)**

(c) Write down the steps followed while freeing a region in Unix. **(7)**

(d) What type of restrictions are there for manipulation of different regions of a process? Justify your answer. **(8)**

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE**.

5. (a) Write the pseudo code of aging algorithm for page replacement. **(5)**

(b) In the aging algorithm for page replacement 8-bits are used for maintaining the counter corresponding to each page. Initially, all the counters are set to zero. Let there are six page frames in the physical memory numbered from 0 to 5. After the first clock tick, the R bits for pages 0 to 5 have the values 101011 (page 0 is 1, page 1 is 0 page 2 is 1, etc.). At subsequent clock ticks, the R bit values are 110010, 110101, and 100010. Give the binary values of the counters after each tick. Which page will be replaced if a replacement is required after the last tick? **(8+2=10)**

(c) Draw a block diagram showing the interaction among translation lookaside buffer, page table and physical memory. **(10)**

(d) Suppose you are using paging technique for implementing virtual memory with 32-bit address space and 4 KB page size. You have decided to keep the entire page table for each process in main memory. As a result, you have to allocate a huge amount of memory for keeping the page tables. Explain with necessary figures how you can solve this problem. You must prove the space efficiency of your technique. Make necessary assumptions and mention those assumptions. **(10)**

6. (a) Consider the following code for the producer-consumer problem using semaphore. **(7)**

```
semaphore mutex = 1 ;
semaphore empty = 5;
semaphore full = 0;

void producer(void)
{
      int item;
      while (TRUE) {
         item = produce_item();
         /*generate something to put
            in buffer*/
         down(&mutex);
         down(&empty);
         insert_item(item);
         /*put new item in buffer*/
         up(&mutex);
         up(&full);
      }
}
```

```
void consumer(void)
{
      int item;
      while (TRUE) {
         down(&full);
         down(&mutex);
         item = remove_ item();
         /*take item from buffer*/
         up(&mutex);
         up(&empty);
         consume_item(item);
         /*do something with the
            item*/
      }
}
```

The above code will work properly - Prove or disprove?

CSE 313

6. (b) Suppose two processes, P1 and P2 are running concurrently in a single processor system. The system contains one printer and one DVD-ROM. P1 has 500 instructions and P2 has 300 instructions. P1 requests the DVD-ROM at 100th instruction and the printer at 200th instruction. P1 releases the DVD-ROM and printer at 300th and 400th instruction, respectively. P2 requests the printer at 50th instruction and the DVD-ROM at 100th instruction. P2 releases the printer and DVD-ROM at 150th and 200th instruction, respectively. Explain the concept of unsafe region and deadlock for this system graphically. Make necessary assumptions and mention those assumptions. **(10)**

(c) Suppose in a particular system, each process needs a fixed amount of memory. Explain with figure how you can allocate disk spaces for storing swapped out pages in the swap area for this system using least space in the main memory. **(5)**

(d) What is the difference between unsafe state and safe state? **(5)**

(e) A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows: **(8)**

	Allocated	*Maximum*	*Available*
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

Find the smallest value of x for which this is a safe state. Make necessary assumptions and mention those assumptions.

7. (a) Suppose in a particular environment with single processor, there are three types of processes: system processes, interactive processes and batch processes. Each system process has a predefined importance level which must be considered. All batch processes have known running time and are submitted at the same time. All interactive processes should be treated with equal importance. 50% of the total CPU time should be allocated to the system processes, 30% of the total CPU time should be allocated to the interactive processes and the rest of the total CPU time should be allocated to the batch processes. Now design a suitable scheduler for this environment. Make necessary assumptions and mention those assumptions. **(8)**

(b) Suppose you are using the linked list allocation for keeping track of which disk blocks go with which file. How can you modify this scheme to ensure faster random access? **(5)**

(c) What are the differences between blocking and non-blocking system call? **(5)**

পা পিছলে পড়ে যাওয়া লজ্জার কথা নয়।
বরং যথা সময়ে উঠে না দাঁড়ানোই লজ্জার ব্যাপার।

Contd ... Q. No. 7

(d) Consider the following workload: **(8+4=12)**

Process	Duration (sec)	Arrival time (sec)
P1	40	50
P2	70	20
P3	50	0
P4	100	10
P5	50	70

Draw the Gantt chart for Round Scheduling with quantum 30 seconds and calculate the average turnaround time. Make necessary assumptions and mention those assumptions.

(e) Why is boot block located in every logical partition? **(5)**

8. (a) Consider a web server which has to serve the multiple requests at all time. The requests come asynchronously and each request takes variable amount of time. You have a processor with 50 cores where 50 threads can be executed concurrently. Now explain with pseudo-code, how you can write the web server program for the above scenario in the most efficient manner. **(6)**

(b) Consider a simple command interpreter, called E_SHELL (like Linux shell) which serves user commands using operating system features. When started, E_SHELL asks for one command from the user. When the user types a command and press enter E_SHELL waits for the command to complete and then asks for the next command. If the user puts an ampersand (&) after a command, E_SHELL does not wait for it to complete. Instead it just asks for the next command immediately. Now write a code in C language for E_SHELL using the system calls shown in the following Figure. Make necessary assumptions and mention those assumptions. **(8)**

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s =execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Figure for Questions 8(b)

(c) What is a TSL instruction? Implement two procedures, enter_region() and leave_region() using TSL to achieve mutual exclusion, where each process has to call enter-region() before entering its critical region and leave_region() after leaving its critical region. What are the drawbacks of this method? **(8)**

(d) Consider a modified dining philosopher problem, where five philosophers are sitting on a round table and every two philosopher, sitting side by side share a fork. The life of a philosopher consists of alternate periods of eating and thinking. When a philosopher gets hungry, if any one of the forks is available, he eats for a while, then puts down the fork, and continues to think. Provide a solution to this problem using interprocess communication methods. Your solution must ensure maximum parallelism. Make necessary assumptions and mention those assumptions. **(13)**

L-3/T-2/CSE **Date : 12/05/2014**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2011-2012

Sub : **CSE 313** (Operating System)

Full Marks: 210 Time : 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this section. Answer any **THREE**.

1. (a) Draw the block diagram of the Unix System Kernel. Briefly list all the major subsystems. **(10)**

   (b) Why is buffer cache used in Unix System? What are the benefits of using it? **(7)**

   (c) Explain a scenario where race for free buffer (cache) may occur. What can be a possible solution of such a race condition? **(10)**

   (d) How are the buffers marked as "delayed write" processed while allocating a buffer in the Unix Kernel? **(8)**

2. (a) Explain three scenarios while allocating a buffer (using getblk algorithm) when the buffer cache of free buffers is non-empty. **(12)**

   (b) Give one example of file access times when a file's content has not been changed recently but its permission bits have been changed. **(6)**

   (c) What additional information are used in in-core inodes compared to disk inodes? Why do we need them in in-core inodes? **(7)**

   (d) Considering the Unix System V inode structure, locate the $4,00,000^{th}$ byte offset in a file. Assume block numbers to explain your answer. **(10)**

3. (a) Write down an algorithm that converts path name to inode. **(10)**

   (b) Draw and explain Unix data structures used to maintain the linked list of free disk blocks. **(10)**

   (c) What is the role of remembered inode in freeing inodes? Give examples to explain your response. **(7)**

   (d) Draw the process state transition diagram for Unix system. **(8)**

4. (a) What happens when the Kernel wakes up a sleeping process? Explain in brief. **(10)**

   (b) Write down an algorithm for duplication of a region. When does such duplication happen in a system? **(10)**

   (c) What happens when open file call is invoked? Explain with respect to file data structures. **(8)**

   (d) Which region's size can be changed using a system call (eg., growreg)? Which cannot be changed? Give examples and justify your answer. **(7)**

Contd P/2

CSE 313

SECTION – B

There are **FOUR** questions in this section. Answer any **THREE**.

5. (a) Suppose you are using paging technique for implementing virtual memory. You have decided to keep the entire page table for each process in main memory. As a result an extra memory lookup is needed for each virtual address generated by the CPU to access the page table. Explain with necessary figures how you can speed up paging. **(13)**

(b) What is inverted page table? Why is it needed? How can you make the search in inverted page table efficient? Give necessary figures. **(5+2+5=12)**

(c) In the aging algorithm, for page replacement, 8 bits are used for maintaining the counter corresponding to each page. Initially all the counters are set to zero. Let there are six page frames in the physical memory numbered from 0 to 5. After the first clock tick the R bits for pages 0 to 5 have the values 101011 (page 0 is 1, page 1 is 0, page 2 is 1, etc.). At subsequent clock ticks the R bit values are 110010, 110101, and 100010. Give the values of the counters after each tick. Which page will be replaced if a replacement is required after the last tick? **(8+2=10)**

6. (a) Define resource deadlock? What are the conditions that must be present for a resource deadlock to occur? **(2+6=8)**

(b) Discuss various ways of recovering from deadlock. **(8)**

(c) Define safe state. What is the difference between unsafe state and deadlock? **(2+2=4)**

(d) Discuss with an example how you can avoid the circular wait condition. **(7)**

(e) A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows: **(8)**

	Allocated	Maximum	Available
Process A	1 0 2 1 1	1 1 2 1 3	0 0 × 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

Find the smallest value of x for which this is a safe sate.

7. (a) "For real time systems having the right answer but having it too late is often just as bad as not having it all all"— Do you agree with this statement? Justify your position. **(5)**

Contd P/3

CSE 313

Contd... Q. No. 7

(b) Write short notes on the following topics **(5+5=10)**

(i) Lottery scheduling

(ii) Convoy effect

(c) Discuss the difference between compute-bound and I/O bound process with figures. **(5)**

(d) Consider the following workload: **(5×3=15)**

Process	Priority (Lowest number has the highest priority	Burst Time (sec)	Arrival Time (sec)
P1	4	40	50
P2	3	70	10
P3	1	50	0
P4	5	100	0
P5	2	50	70

Draw the time scale diagram and calculate the average turnaround for the following scheduling algorithms:

(i) Shortest Job First

(ii) Shortest Remaining Time Next

(iii) Round Robin with quantum 30 sec

8. (a) Which of the following are "per process item", which are "per thread item", and which are neither? **(8)**

Program Counter, Stack, Address Space, Global Variables, Registers, Open Files, Signals, Local Variables

(b) Explain with figures how you can dynamically allocate disk spaces for storing paging in the swap area. **(8)**

(c) Draw the models of main memory presented to the programmer by paging scheme and segmentation scheme. **(2.5+2.5=5)**

(d) Write short notes on the following topics **(5+5=10)**

(i) Semaphore

(ii) Monitor

(e) Explain the use of barriers with necessary figures. **(4)**

A.S.M. Sohidull Islam

L-3/T-2/CSE Date : 17/12/2012

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2010-2011

Sub : **CSE 313** (Operating System)

Full Marks: 210 Time : 3 Hours

USE SEPARATE SCRIPTS FOR EACH SECTION

The figures in the margin indicate full marks.

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE**.

1. (a) Given that the maximum file size of combination of direct, single indirection, double indirection and triple indirection in an inode-based filesystem is approximately the same as a filesystem solely using triple indirection. Then why do not simply use only triple indirection to locate all file blocks? **(10)**

   (b) What is the maximum file size supported by a file system with 16 direct blocks, single, double, and triple indirection? Here the block size is 512 bytes and disk block numbers can be stored in 4 bytes. Locate the middle data block. **(7+8=15)**

   (c) How many pointers are there in a Buffer Header? Shortly describe the necessity of the pointers? **(10)**

2. (a) What is reference count and link count of an inode? [see iput algorithm in Fig. 2] **(10)**

   (b) Discuss a system implementation that keeps track of free disk blocks with a bit map instead of a linked list of blocks. What are the advantages and disadvantages of this scheme? **(15)**

   (c) Suppose you have a 700 MB CD-ROM. Do you think it is possible to store a 700 MB file in this CD-ROM? Explain your answer. **(10)**

3. (a) To maintain atomicity of buffer cache, we raise or lower processor execution level. Is it possible to avoid raising/lowering execution level using lock? If yes, describe how would you apply it for scenario 5 in getblk algorithm. [see getblk algorithm in Fig. 1.] **(10)**

   (b) Which part of getblk algorithm is responsible to maintain Least Recently Used structure in Buffer Cache? [see getblk algorithm in Fig. 1] **(7)**

   (c) Why is there the while loop in getblk algorithm? [see getblk algorithm in Fig. 1] **(8)**

   (d) Why size of a disk block is a critical issue? What are the problems if it is too large or too small? **(10)**

4. (a) Compare "Linked list allocation file system" and "Linked list allocation using a table in memory file system". **(12)**

   (b) How free disk blocks are managed in the free disk block list of a super block? **(13)**

   (c) Suppose all the processes are in "asleep" state. How it is possible that a process will wakeup and come to "ready to run" state? **(10)**

Contd P/2

CSE 313

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE**.

5. The "Search-Insert-Delete" problem can be stated as follows:

Three kinds of threads share access to a file: *Searchers*, *Inserters* and *Deleters*. *Searchers* merely examine the list; hence they (Searchers) can execute concurrently with each other. *Inserters* add new items to the end of the list; insertions must be mutually exclusive to preclude two inserters from inserting new items at about the same time. However, one insert can proceed in parallel with any number of searches. Finally, *deleters* remove items from anywhere in the list. At most one *Deleter* process can access the list at a time, and deletion must also be mutually exclusive with searches and insertions.

(a) Write pseudocodes for Searchers, Inserters and Deleters that enforce this kind of three-way categorical mutual exclusion mentioned above using Semaphore and/or Mutex. **(27)**

(b) What do you mean by starvation? Does your solution of the "Search-Insert-Delete" problem result in any starvation? Briefly explain. **(3+5=8)**

6. (a) Write down two advantages of Micro-kernel architecture. **(4)**

(b) Write short notes on **(5×3=15)**

(i) Remote Procedure Call (RPC)

(ii) Upcall in Scheduler Activation

(iii) System call implementation of Finite State Machine thread model.

(c) Given the following state for the Banker's Algorithm. There are 6 processes P0 through P5 and 4 resource types: A (15 instances); B (6 instances), C(9 instances); D (10 instances). For the following current allocation and maximum need, should a new request (3, 2, 3, 3) from P5 be granted? **(10)**

Current Allocation				
Process	A	B	C	D
P0	2	0	2	1
P1	0	1	1	1
P2	4	1	0	2
P3	1	0	0	1
P4	1	1	0	0
P5	1	0	1	1

Maximum Need				
Process	A	B	C	D
P0	9	5	5	5
P1	2	2	3	3
P2	7	5	4	4
P3	3	3	3	2
P4	5	2	2	1
P5	4	4	4	4

(d) What is memory segmentation? What are the benefits of using multiple segments? **(3+3=6)**

Contd P/3

CSE 313

7. (a) Explain the architecture of a typical virtual machine. Write down some disadvantages of this approach. **(7)**

(b) What are the disadvantages of implementing threads in kernel space solely? **(6)**

(c) Verify whether the following code snippet solves the critical section problem for a two process environment. Two processes, say Process 1 and Process 2, running infinitely call **enter_critical_section** and **leave_critical_section** procedures with their corresponding ids before entering and after leaving the critical section respectively. **(15)**

```
Boolean blocked [2];
Int turn;

void enter_critical_section(int process_id)
{
    blocked[process_id] = true;
    while(turn != id) {
        while(blocked[1 - process_id]) {
            //do nothing
        }
        turn=id;
    }
}
void leave_critical_section(int process_id)
{
    blocked[id] = false;
}
```

(d) Write down the address translation procedure for inverted page table scheme with its advantages. **(7)**

8. (a) Write down the conditions for occurrence of deadlocks. **(8)**

(b) Consider the following workload: **(5×3=15)**

Process	Priority (Lowest number has the highest priority)	Burst Time (sec)	Arrival Time (sec)
P1	4	40	50
P2	3	70	10
P3	1	50	0
P4	5	100	0
P5	2	50	70

Draw the Time scale diagram and calculate the average waiting time for the following scheduling algorithms:

(i) Non-preemptive Shortest Job First
(ii) Shortest Remaining Time First
(iii) Round Robin with quantum 30 sec.

(c) Consider the following string of page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1. List the total number of page faults for the following page replacement strategies with frame size = 3 **(12)**

(i) Optimal page replacement algorithm
(ii) First-in-first-out page replacement algorithm

```
algorithm getblk
input:   file system number
         block number
output: locked buffer that can now be used for block
{
        while (buffer not found)
        {
                if (block in hash queue)
                {
                        if (buffer busy)        /* scenario 5 */
                        {
                                sleep (event buffer becomes free);
                                continue;       /* back to while loop */
                        }
                        mark buffer busy;       /* scenario 1 */
                        remove buffer from free list;
                        return buffer;
                }
                else      /* block not on hash queue */
                {
                        if (there are no buffers on free list)       /* scenario 4 */
                        {
                                sleep (event any buffer becomes free);
                                continue;       /* back to while loop */
                        }
                        remove buffer from free list;
                        if (buffer marked for delayed write) {     /* scenario 3 */
                                asynchronous write buffer to disk;
                                continue;       /* back to while loop */
                        }
                        /* scenario 2 -- found a free buffer */
                        remove buffer from old hash queue;
                        put buffer onto new hash queue;
                        return buffer;
                }
        }
}
```

Fig 1. getblk algorithm

```
algorithm iput              /* release (put) access to in-core inode */
input:   pointer to in-core inode
output: none
{
        lock inode if not already locked;
        decrement inode reference count;
        if (reference count == 0)
        {
                if (inode link count == 0)
                {
                        free disk blocks for file (algorithm free, section 4.7);
                        set file type to 0;
                        free inode (algorithm ifree, section 4.6);
                }
                if (file accessed or inode changed or file changed)
                        update disk inode;
                put inode on free list;
        }
        release inode lock;
}
```

Fig 2. iput algorithm

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2009-2010

Sub : CSE 313 (Operating Systems)

Full Marks: 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE**.

1. (a) What are the four conditions of deadlock? How do you avoid them? **(12)**

(b) Multiple processes need to run on a non-preemptive system. Process A is currently active but needs to acquire an exclusive lock on a certain resource R in order to continue. Unfortunately, Process B currently has an exclusive lock on R. Process A can spin-wait until R becomes available, or A can yield so another waiting process may run. State which of the following statements are correct/incorrect, and provide justification for your answer. **(12)**

I. If this is a uni-processor machine, then A should yield to another process.

II. If this is a multi-processor machine, but B is inactive, then A should yield to another process.

III. If this is a multi-processor machine, and B is active, and the time to complete a context switch does not exceed the time that B will retain the lock, then A should yield to another process.

IV. If this is a multi-processor machine, and B is active, but the time to complete a context switch exceeds the time that B will retain the lock, then A should spin-wait.

(c) Explain 3 methods for passing parameters to a system call. **(6)**

(d) What are the advantages and disadvantages of a Microkernel? **(5)**

2. (a) While designing a kernel, you must decide whether to support kernel-level or user-level threading. State which of the following statements are true or false, and provide justification to your answer. **(9)**

I. Kernel-level threading may be preferable to user-level threading because storing information about user-level threads in the process control block would create a security risk.

II. User-level threading may be preferable to kernel-level threading because in user-level threading, if one thread blocks on I/O, then the process can continue.

III. User-level threading may be preferable to kernel-level threading because in user-level threading, less expensive overhead is required when one thread blocks and another begins to run instead.

Contd P/2

(b) A single-CPU computer has an operating system with a purely priority driven pre-emptive scheduler that runs three processes. Process A is a low-priority process that runs once per hour; when it runs, it acquires an exclusive write lock on a log file, writes the current position and speed to the log file, closes the file, and then exits. Process B is a medium-priority video-streaming process that usually isn't running; when it runs, it does so continuously for an arbitrary length of time. Process C is a high-priority navigation process that runs once per minute; when it does, it acquires an exclusive write lock on the log file, issues some commands to the rudder, closes the file, and then exits.

State which of the following statements are true or false. Provide justification to your answer. **(9)**

I. Starvation may occur for one or more processes for an arbitrary length of time.

II. Priority inversion may occur for an arbitrary length of time.

III. Deadlock may occur.

(c) Explain, how an operating system: **(5+5)**

i. Protects itself from user programs. Provide necessary illustrations.

ii. Protects one program from accessing other program's data. Provide necessary Illustrations.

(d) What is a "Test and Set" instruction? Implement the Lock_Acquire () and Lock_Release() functions using "Test and Set" instruction, which minimizes the busy waiting time. **(7)**

3. (a) Consider the following code, which prints 100 integers. The queue Q is not thread-safe, hence the code uses semaphore smx to assure thread-safety. Queue Q provides two functions, add () and remove(), for first-in-first-out loading and unloading of items from a list. The program will crash if remove() is called while the queue is empty. **(12)**

```
function loadMe () {
    for ( int x = 0 ; x < 100;  x = x + 1 )   {
        Q.add ( x)
        V( smx )
    }
}
function printMe () {
    while ( true ) {
        P( smx)
        int value = Q. remove () ;
        print (value ) ;
    }
}
```

One thread executes loadMe() at an arbitrary time, while another thread executes printMe() at another arbitrary time. State which of the following statements are true or false, and provide justification to your answer.

i. If the V operation implements non-blocking atomic increment, and the P operation implements blocking atomic decrement, then the code above will work properly.

ii. The code above may crash if multiple threads simultaneously execute printMe(), even if only a single thread executes loadMe().

iii. The code segment above is functionally equivalent to the following, where mx is a mutex.

```
Function loadMe()  {
    for ( int x = 0; x < 100; x = x + 1) {
        lock ( mx )
        Q. add ( x )
        unlock ( mx )
    }
}
Function printMe()  {
    while ( true )  {
         lock( mx )
         int value = Q. remove ()
         print (value )
         unlock (mx )
    }
}
```

(b) Prove the correctness of Peterson's Solution to critical section problem. **(7)**

(c) Five batch jobs A through E arrive at a computer center at almost the same time. They have estimated running times of 10, 6 2, 4 and 8 minutes. Their priorities are 3,5,2,1, and 4, respectively, with 5 being the highest priority. **(16)**

For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

I. Round Robin

II. Priority Scheduling

III. FCFS (Run in order 10, 6, 2, 4, 8)

IV. SJF

For I (Round Robin), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For II-IV, assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

Contd P/4

4. (a) While designing a preemptive job scheduling subsystem, an operating system manufacturer selected Round-Robin. State which of the following requirements will be satisfied or not satisfied. If the requirement is not satisfied then suggest an algorithm that may satisfy it. **(10)**

   I. The subsystem must minimize the number of context switches.

   II. The subsystem must achieve optimal throughput of jobs.

   III. The subsystem must guarantee the starvation will not occur.

   IV. The subsystem must guarantee that if job J1 arrives before job J2, then J1 finishes before J2.

   (b) What are the pros and cons of Programmed I/O and DMA driven I/O? **(6)**

   (c) What are the functionalities of the top half and bottom half portions of a device driver? Differentiate between Block Devices and Character Devices. **(6+5)**

   (d) Which of the following are "per process item", which are "per thread item", and which are neither? **(8)**

   Program Counter, Stack, Address Space, Global Variables, Registers,, Open Files, Signals and Local Variables.

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE**.

5. (a) Consider a memory management scenario where the virtual address space of each process is divided into three segments (code, data and stack). Each segment has its own page table. The i^{th} entry in any segment's page table maps that segment's i^{th} virtual page number to a physical page number. Each process maintains a segment table which contains the starting virtual address of that segment. Given the segment table along with page tables in Figure 4(a) what is the physical address corresponding to the 32 bit virtual address $00009C7F_{16}$. (Page Size = 4K) **(7)**

   (b) Consider a paging scheme with Two level page tables and 4K page size. The first 12 bits of 32 bit virtual address in an index to the first level page table, the next 8 bits are index to the second level page tables. Consider a process which needs 12 megabytes from its address space. The bottom 4 MB for stack, the next 4 MB for data and the top 4 MB for its code / text. Calculate the percentage of space savings using this scheme over a single level page table. **(10)**

   (c) In the aging algorithm for page replacement 16 bits are used for maintaining the counter corresponding to each page. Initially all the counters are set to 0. Let there are 8 pages in the physical memory and the pages are referenced in the order 1 2 0 5 4 7 8 6 3 2 1 4 5 0 0 0 2 1 4 5 6 2 1 7 5 2 3 5 7 2 5 8 7 0 1 2 3 6 5 4 1 2 3 5 4 7 8 3 6 3 2 1 4 5 5 6 8 7 7. After this sequence of pages is referenced, which page will be replaced if a replacement is required? **(12)**

Contd P/5

(d) If the average size of programs running in a system is 2 MB and each page table entry is 16 bytes then calculate the optimal page size for paging. **(6)**

6. (a) Draw a block diagram showing the kernel architecture of UNIX. **(12)**

(b) Draw the process state transition diagram for UNIX process. **(12)**

(c) Draw the layout of a logical disk partition in UNIX File system and give description of different components. **(11)**

7. (a) In the buffer cache mechanism UNIX follows LRU (Least Recently Used) policy for replacing a buffer. Modify the insertion / removal policy in the buffer free list such that kernel follows **(9+9=18)**

(i) FIFO policy for buffer replacement

(ii) Least frequently used policy for buffer replacement

(b) Consider the UNIX file system where the Inodes have 10 direct entries, 1 single indirect block, 1 double indirect block and 1 triple indirect block. Locate the byte offset 8589934592 in a file. **(8)**

(c) How UNIX kernel handles 3 nested levels of interrupt / system calls? Describe with necessary figures showing the status of dynamic system level context at different levels of nesting. **(9)**

8. (a) Normally the file systems that use Inode, put the Inodes near the beginning of the disk. In such arrangements the average distance between Inodes and its data blocks is about half of the number of tracks (a physical disk is divided into concentric circles called tracks), which causes longer seek time. What technique(s) can be used to minimize this distance so that the seek time is reduced? **(7)**

(b) Write short note on the following topics: **(3×4=12)**

(i) FAT

(ii) Contiguous disk block allocation

(iii) Different file structures

(c) Consider two threads A and B, and two semaphores X and Y. The two threads obtain and release the semaphores in the following order: **(8)**

Thread A	Thread B
X.P()	Y.P()
Y.P()	X.P()
Y.V()	X.V()
X.V()	Y.V()

Is there any sequence of obtaining / releasing the semaphores which will cause A and B to be deadlocked? If such sequence exists then write the sequence and correct the orders of semaphore obtain / release such that deadlock can never occur. If such sequence does not exist then explain why it does not exist.

(d) Consider a system with four resources R_1, R_2, R_3, and R_4 with 1, 1, 2 and 4 instances respectively. The following table shows the requests made by thread A, B and C for the resources. **(8)**

A	R_1
B	R_2
C	R_3

The following table shows the allocation of resources to the threads A, B, C

	A	B	C
R_1	0	1	0
R_2	0	0	1
R_3	1	1	0
R_4	0	0	0

Detect if there is any deadlock in the system. if a deadlock is detected then which threads are in deadlock?

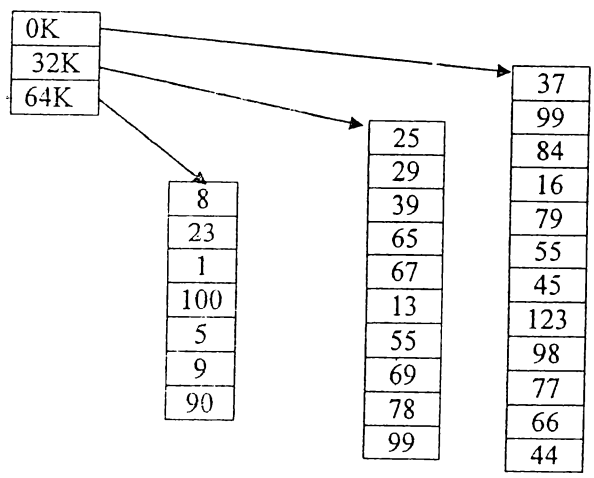


Figure 4(a): Segment Table and Page Table

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L–3/T–2 B. Sc. Engineering Examinations 2008-2009

Sub : **CSE 313** (Operating System)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE.**

1. (a) What is the advantage of taking page size in powers of 2? **(5)**

   (b) Consider a virtual address ABCDDCBA in hex, and a page size of 16 KB. Calculate for the virtual address: **(2 ½+2 ½=5)**

   (i) Page table index,

   (ii) Page offset.

   (c) Define : **(4×2=8)**

   (i) Demand Paging,

   (ii) Working Set,

   (iii) thrashing,

   (iv) Memory Compaction.

   (d) How binary search trees can be used for the following memory management scenarios? **(17)**

   (i) Locating free space and used space,

   (ii) Loading a process into memory,

   (iii) Termination of a process.

   If it is impossible to accomplish the above three tasks then justify your negative answer.

2. (a) Write the ways to pass parameters between a user program and the operating system. **(9)**

   (b) Write short notes on: **(4×6=24)**

   (i) Blocking system call,
   (ii) Non blocking system call,
   (iii) User level thread,
   (iv) Kernel level thread.

   (c) Define pop up thread. **(2)**

3. Consider the following set of processes:

Process	Burst time (ms)	Priority
1	3	3
2	10	1
3	1	3
4	5	4
5	2	2

The processes are assumed to have arrived in the order 5, 2, 1, 4, 3.

15

Contd .....:P/2

(a) Draw three Gantt charts that illustrate the execution of these processes using the following scheduling algorithm: **(3×6=18)**

(i) First Come First Serve,

(ii) Shortest Job First,

(iii) Round Robin.

(b) What is the turnaround time for each process for each of the algorithms? **(3×3=9)**

(c) What is the waiting time for each process for each of the algorithms? **(8)**

4. (a) What are the requirements for the solution of the critical section problem? **(10)**

(b) What are the differences between mutex and semaphore? **(7)**

(c) Consider the dining philosopher problem, where five philosophers are sitting on a round table and every two philosophers sitting side by side share a fork. A philosopher eats only if he acquires both the forks. When the fork is not available, he waits. Now consider a variant of the problem where a philosopher will eat if any one of the forks is available. Provide a solution to this problem using synchronization primitives. Make sure that your solution satisfies the requirements of the solution of critical section problem. **(18)**

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE**.

5. (a) Draw a block diagram showing the kernel architecture of UNIX OS. **(15)**

(b) Write short notes on the following: **(3×4=12)**

(i) Boot block,

(ii) Super block,

(iii) Inode List,

(iv) Data Block.

(c) "Executing a system call in UNIX does not make a context switch" – justify this statement. **(6)**

(d) Write the names of different modes of UNIX process. **(2)**

6. (a) Draw a diagram showing the possible states of an UNIX process and the possible transitions between them. **(15)**

(b) Describe the way process context is saved and restored when an interrupt occurs. **(12)**

(c) Write the static and dynamic contexts of a process. **(4+4=8)**

Contd P/3

CSE 313

7. (a) Describe a scenario when inconsistency may arise in the hash queue for not locking it during its modification. **(12)**

   (b) Read/Write from disk in UNIX is done in units of blocks. Then why does UNIX provide support for character files? **(8)**

   (c) Describe how Least Recently Used algorithm is used for buffer replacement. **(12)**

   (d) What is the necessity of event descriptor field in the process table entry? **(3)**

8. (a) What is the maximum size of a file if the inode table of contents has 16 direct blocks, 1 single indirect block, 1 double indirect block, 1 triple indirect block and 1 quadruple indirect block? Assume that a logical block holds 1 KB and the block numbers are accessed by 32 bit integers. **(5)**

   (b) Consider an inode with 10 direct blocks, 1 single indirect, 1 double indirect and 1 triple indirect block. Assume that block size is 2 KB. Then, which entry of the inode table of contents can locate the disk address 1 MB? **(8)**

   (c) When is an inode removed from the incore list of inodes? **(4)**

   (d) Why is boot block located in every logical partition? **(3)**

   (e) The processes in UNIX consider files as a stream of bytes, but the file data is kept in the disk in a scattered way. How does UNIX handle this? Describe the necessary data structure UNIX uses for handling this and also describe how a process reads / writes bytes into a file. **(15)**

Date : 06/08/2009

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L-3/T-2 B. Sc. Engineering Examinations 2007-2008

Sub : **CSE 313** (Operating System)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **NINE** questions in this Section. Answer any **SEVEN**.

1. (a) How does FAT file system keep track of disk blocks of files? How much memory will be needed to keep the file allocation table always in memory for a 64 GB hard disk and 4 KB block size? **(8)**

   (b) What is hard link and soft link? How are they implemented in a file system? **(7)**

2. (a) How does kernel anticipate the need of 2nd disk block when a process reads a file sequentially? **(9)**

   (b) Why is the buffer released after finding the second block in buffer cache in block read ahead algorithm? **(6)**

3. (a) Write down the components of the context of a process. **(7)**

   (b) How is context layer updated in case of context switch? How is it different from system call? **(8)**

4. (a) In an extended inode system, an entire block is assigned to each inode. The remaining portion of the block is used to store data. If block size is 2 KB and each block address is 32 bits, what is the maximum file size? Assume classic inode structure and inode header size is 256 bytes. **(6)**

   (b) Why does kernel perform double checking when kernel wants to assign an inode to a new file? Illustrate the problem with an example in absence of double checking. **(9)**

5. (a) Why are the contents of in-core inode and disk inode different? State those contents that are only present in in-core inode. **(6)**

   (b) Briefly explain the differences between inode and buffer cache mechanism. **(9)**

6. (a) What is the significance of 'remembered inode'? Give an example when disk contains a free inode whose number is less than remembered inode? **(5)**

   (b) What happens when super block free disk list is already full and a disk block is released? **(5)**

   (c) What happens when super block free disk list contains only one disk block and kernel wants to assign a new block? **(5)**

7. (a) How does kernel map virtual address to physical address using page table? Assume starting address of text, data, and stack region of a process is 8K, 32K, and 64K. The sizes of these regions are 8 KB, 16 KB, and 32 KB respectively. What happens when the process wants to access following virtual addresses? (Assume page size 1 KB) **(9)**

   (i) 64K (ii) 40K (iii) 24K

   (b) How does kernel access the U area of the currently running process using same virtual address? **(6)**

8. (a) Draw the process state transition diagram of UNIX operating system. **(9)**

   (b) How is the first process created in UNIX? State its functions. **(6)**

9. Illustrate the kernel data structure of file management when the following system calls are executed: **(15)**

   (i) Process OS1 opens a file named "/home/ysu" in read only mode that returns the descriptor fd1.

   (ii) Process OS2 opens a file named "/home/ahr" in write only mode that returns the descriptor fd2.

   (iii) Process OS3 opens the same file as process OS1 in write only mode that returns the descriptor fd3.

   (iv) Process OS2 duplicates fd2 and gets a new descriptor fd4.

   (v) Process OS2 writes 1 KB data using fd2.

   (vi) Process OS1 and OS3 gets chance and reads and writes 1 KB data alternatively for two times.

   (vii) Process OS2 closes fd2.

   (viii) Process OS2 writes 1 KB data using fd4.

CSE 313

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE**.

10. (a) Why relocation and protection are needed in multiprogramming? Write a strategy that solves both the relocation and protection problem. **(7)**

(b) Explain with example the Belady's anomaly. What is the property that algorithms must have not to suffer from this anomaly? **(8)**

(c) What is working set concept? Describe a page replacement algorithm based on the working set. **(2+10=12)**

(d) A computer has four page frames. The time of loading, time of last access and the R and M bits for each page are shown below (the times are in clock ticks) : **(8)**

Page	Time of Loading	Time of last Referenced	R	M
0	75	125	0	0
1	18	110	1	0
2	10	212	1	1
3	105	220	1	1

(i) Which page will be replaced by NRU?

(ii) Which page will be replaced by FIFO?

(iii) Which page will be replaced by second chance?

(iv) Which page will be replaced by LRU?

11. (a) What are the general strategies to deal with deadlock? Which of these strategies is followed by most operating systems and why? **(7)**

(b) What are the ways to attack the circular wait condition? **(7)**

(c) Differentiate between starvation and deadlock. **(4)**

(d) Assume a system with three resource types, C = <10, 5, 7> (this is the total number of resources in the system, and not what is currently available). The current allocation and maximum need matrix are shown below – **(4+4+4+2=14)**

Current Allocation			
Process	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Maximum Need			
Process	R_0	R_1	R_2
P_0	7	5	3
P_1	3	2	2
P_2	9	0	2
P_3	2	2	2
P_4	4	3	3

Contd P/4

(i) Draw a resource wait graph corresponding to this snapshot of the system.

(ii) Show that the system is currently is safe state.

(iii) Suppose now process P_1 requests one instance of resource type R_0 and two instances of type R_2 so, request = (1, 0, 2). Determine whether this request can be granted using Banker's Algorithm.

(iv) If instead of (iii), a request of (4, 2, 2) by P_4 is made, determine whether this request can be granted.

(e) Why is Banker's algorithm not used practically to avoid deadlocks? **(3)**

12. (a) What is a context switch? "Thread context switch is lighter than process context switch" – do you agree or disagree? Justify with reasons. **(7)**

(b) Why two kinds of semaphores are needed in the critical section of producer-consumer problem? **(5)**

(c) Compare user level threads with kernel level threads. Which one is better? **(7)**

(d) Describe the gang scheduling approach. When is it mainly used? **(6)**

(e) Five Batch jobs A through E arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4 and 8 minutes respectively. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead. **(10)**

(i) Round robin (assume each job gets fair share of CPU)

(ii) First come first served (run in order A to E)

(iii) Shortest job first.

13. (a) 'Determining the best page size requires balancing several competing factors' – what are those factors? Based on these, derive the formula for optimum page size. **(6+3=9)**

(b) Write short note on microkernel structure. **(5)**

(c) How can performance of a web server be enhanced using threading? **(6)**

(d) What is TLB? How paging can be speeded up using the TLB. **(8)**

(e) Consider a swapping system in which memory consists of the following hole sizes in memory order : 10 KB, 4 KB, 20 KB, 18 KB, 7 KB, 9 KB, 12 KB and 15 KB. Which hole is taken for successive segment requests of 12 KB, 10 KB and 9 KB if **(7)**

(i) first fit

(ii) best fit

(iii) worst fit

(iv) next fit is used for allocating the three segments.

Apply same fit technique to all 3 segment requests in each case.

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA

L–3/T–2 B. Sc. Engineering Examinations 2006-2007

Sub : **CSE 313** (Operating System)

Full Marks : 210 Time : 3 Hours

The figures in the margin indicate full marks.

USE SEPARATE SCRIPTS FOR EACH SECTION

SECTION – A

There are **FOUR** questions in this Section. Answer any **THREE.**

1. (a) What are the purposes of using boot block, super block, inode list and data blocks in Unix System? **(12)**

   (b) How does file subsystem interact with hardware in Unix? Explain with respect to Unix System Kernel structure. **(8)**

   (c) Write down the algorithm for block read ahead. **(7)**

   (d) Explain the buffer allocation strategy when there is buffer available in the free list. Use necessary sketch to clarify your idea. **(8)**

2. (a) Compare four different strategies of storing file contents. **(8)**

   (b) What are the major information kept in disk inodes and incore inodes? Why are they different? **(10)**

   (c) Write down the algorithm for allocation of incore inodes. **(10)**

   (d) How to find out the block number and byte offset of an inode? If the starting block number of inode list is 7, number of inodes per block is 16 and size of each disk inode is 72 bytes, them find out the block number and byte offset of inode number 662. **(7)**

3. (a) What happens when the inode reference count becomes zero while releasing an inode from main memory? What if the inode link count also becomes zero? **(7)**

   (b) What is the basic difference in arrangement of superblock free inode list and super block free inode list and super block free disk block list? Explain with figure. **(7)**

   (c) Explain the steps required while reading a file. Use necessary sketch to clarify your idea. **(7)**

   (d) Explain with figure the system-level context of a process. **(7)**

   (e) Consider that in a Unix system, each data block size is 4K bytes and 32 bits are required for data block addressing. Assuming usual table of content structure of Unix system, find out the maximum file size of such system. **(7)**

4. (a) Explain Unix process state transition diagram. **(10)**

   (b) What are the major information kept in process table entry and u-area? **(8)**

   (c) Write down the algorithm to load a portion of a file into a region. **(7)**

   (d) Explain the algorithm for sleep system call. **(10)**

SECTION – B

There are **FOUR** questions in this Section. Answer any **THREE.**

5. (a) Why is a separate swap space needed in a system? Consider a new system which will run only those processes that can grow dynamically at run time. Which strategy do you recommend to follow in case of disk swap area management? Justify your answer. **(4+6=10)**

   (b) Explain the 'working set' concept. Why does the set change faster initially and slower later with respect to time? **(10)**

   (c) Sarwar and Muzahid, two members of a development team are arguing on paging mechanism for their upcoming version of operating system. With 32 bit address space, 4 KB page and 1024 MB RAM, Sarwar argues in favor of multilevel page table. On the otherhand, Muzahid, prefers inverted page table. Whom do you support and why? **(10)**

   (d) Mr. A has just received a memory of size 1GB that is to be connected to his computer. Operating system keeps track of memory usage using bitmaps. What is the size of bitmap in case of 4 KB allocation unit? **(5)**

6. (a) In a multiprogramming system, each job gets fair share of CPU. Consider four jobs arrive at the system on 0, 10, 15, 20 sec. Their running times are 4, 3, 2, 2 sec respectively. Draw the chart of the completion time of the processes. Also compute average turn around time system. **(12)**

   (b) Round robin schedulers normally maintain a list of all runnable processes, with each process occurring exactly once in the list. What would happen if a process occurred twice in the list? Can you think of any reason for allowing this? **(5)**

   (c) Why is spooling used in case of dedicated I/O devices like printer? **(5)**

   (d) How does the system recover bad sector in the disk? Describe any one strategy. **(8)**

   (e) Why does the PCB (process control block) need to record the resources currently in use by a process? **(5)**

7. (a) Three processes share 4 resource units from the same resource class that can be reserved and released only one at a time. This can be done in any order, i. e. we do not have total order imposed on requests. Each process needs a maximum of 2 units. Can we have a deadlock in the system? Explain your answer. **(7)**

(b) What are the four conditions for deadlock? **(6)**

(c) Assume a system with three resource types, C = <10, 5, 7> (this is the total number of resources in the system, and not what is currently available), and the current allocation matrix and maximum claim are shown below. **(12)**

Current Allocation Matrix			
Process	R_0	R_1	R_2
P_0	0	1	0
P_1	3	0	2
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Maximum Claim Table			
Process	R_0	R_1	R_2
P_0	7	5	3
P_1	3	2	2
P_2	9	0	2
P_3	2	2	2
P_4	4	3	3

Draw a resource wait graph corresponding to this snapshot of the system. Using Banker's algorithm, determine whether the state is safe or not. If it is unsafe, enumerate the processes that may get involved in a deadlock. Otherwise, give the sequence in which processes can be run.

(d) Surprisingly, most of the operating systems do not take any step to deal with deadlock. Do you agree with it? If you agree, give reasons in favor of their decision. Otherwise, suggest an alternative strategy that should be followed in the system. **(10)**

8. (a) Consider a ticket-based waiting-queue management system. Each customer arrives at the service centre for a particular service and each service is served at a particular desk. Each customer who wants to access the service desk must get a waiting ticket from a single (shared) ticket issuing desk. Each ticket contains ticket number and the type of service. At a service desk, only one customer can be served at a time. When a customer leaves, he will increment a globally visible counter that indicates the currently active ticket number. Only the customer whose waiting ticket number equals this counter value will be served next. The others have to wait until their ticket number becomes active.

Sketch a realization of this mechanism based on the following two shared counters, which are incremented according to above scenario. **(15)**

```
int ticket; //next ticket to be given out, initially 1.
int active; // currently active ticket, initially 1.
```

Contd ... Q. No. 8(a)

in this system, each customer is represented by one thread, which executes GetTicket when it arrives at the centre.

```
void GetTicket (int serviceId){
        myTkt = AssignTicket ();
        Get service (myTkt, serviceId);
}
```

Write pseudocode for AssignTicket() and GetService(). Use semaphore as the synchronization primitive.

(b) What are the conditions to be satisfied by a solution for race condition? **(5)**

(c) How are the wait/signal operations for monitor different from those for semaphores? **(5)**

(d) The first correct solution for synchronization problem was proposed by Dekker. He only considered two processes in the system. Why does his solution satisfy the conditions of a good solution? For reference, Dekker's solutions is given below **(10)**

Dekker solution:

The two processes P_0 (i) and P_1 (j) share the following variables.

```
var flag: array [0. . 1] of boolean
turn : 0 ..1.
CSEnter (int i)
{
  inside [i] = true;
  while (inside [j])
  {
   if (turn = = j)
   {
     inside [i] = false;
     while (turn = = j ) continue;
     inside [i] = true;
  }
```