UUCMS No. | | | | | | | | | | | |

B.M.S COLLEGE FOR WOMEN, AUTONOMOUS

BENGALURU – 560004

SEMESTER END EXAMINATION – MARCH/APRIL- 2023

B.C.A – III Semester

OPERATING SYSTEMS
(NEP Scheme 2021-22 onwards)

Course Code: BCA3DSC07 **QP Code: 3030**
Duration: 2 ½ Hours **Max. Marks: 60**

Instructions: Answer all the sections

SECTION - A

I. Answer any TEN questions. Each question carries TWO marks. (10X2=20)

1. Define an operating system. Give any two examples of operating system.
2. Differentiate between process and program.
3. Define critical section problem.
4. What is process scheduling?
5. What is starvation? How it is avoided?
6. What do you mean by compaction?
7. What is demand paging?
8. List the advantages of direct access method.
9. Define seek time and latency time.
10. What are boot blocks and bad blocks?
11. Mention the different operations performed on files.
12. Write any four features of Linux system.

SECTION - B

II. Answer any SIX questions. Each question carries FIVE marks. (5X6=30)

13. Explain different types of operating system.
14. What are system calls? Explain different types of system calls.
15. What are semaphores? Explain solution to producer-consumer problem using semaphores.
16. Explain CPU scheduling criteria.
17. What is deadlock? Explain the necessary conditions of deadlock.
18. Define page fault. Explain with neat diagram how to handle page fault.
19. Explain contiguous file allocation method. Mention its advantages and disadvantages.
20. Explain distributed file system.

SECTION - C

III. Answer any ONE question. Each question carries TEN marks. (10X1=10)

21. Consider the following four processes with length of CPU burst time given in milliseconds.

Process	CPU burst time
P1	21
P2	3
P3	6
P4	2

Using SJF and Round Robin scheduling algorithms (Time quantum=5 Ms) find out the average waiting time.

22. Explain Banker's algorithm for deadlock avoidance.

23. Consider the page reference string 1,2,1,0,3,0,4,2,4 with 3 frames, find the number of page faults using FIFO and LRU page replacement algorithms.

*****