**BRAC UNIVERSITY**

**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| Examination: Mid Term  Duration: 1 Hour 10 minutes | Semester: Spring 2022  Full Marks: 30 |

CSE 321: Operating Systems

Answer the following questions.

Figures in the right margin indicate marks.

|  |  |  |
| --- | --- | --- |
| 1.  **CO1** | 1. **Explain** how the two modes of the hardware enable the operating system to securely control user processes. 2. **Explain** the differences between Multiprogramming and Multiprocessing with examples. | [2]  [2] |
| 2.  **CO2** | 1. **Describe** what is the process control block, its contents, and how it is used. In particular, describe its role in context switching.      1. **Find** the output of the following code snippet.  |  | | --- | | int main()  {  int pid1, pid2;  pid1 = fork();  **if** (pid1 == 0) {  pid2=fork();  **if**(pid2 == 0) printf("Hello!\n");  **else**{  wait(NULL);  printf("World!\n");  }  } **else** {  wait(NULL);  printf("Missed Me?\n");  }  printf("Don't miss me!\n");  **return** 0;  } | | [2+1]  [3] |
| 3.  **CO4** | 1. Suppose, in a system, you can use up to 4 processors for 40% of the applications, which means 40% of the applications can run in parallel. **Calculate** the speedup if you increase the number of processors from 1 to 4. 2. Remember that *pthread\_create(tid, NULL, fn, arg)* creates a new thread that executes the function *fn* with the argument *arg*, and *pthread\_join(tid, NULL)* let the current thread wait for the thread with id = *tid* to complete execution. With this information in mind, **find** all possible outputs of the following program.      |  | | --- | | int[] matrix = {4, 6, 9, 2, 5, 3, 0, 1, 11, 13, -1, 7};  void main() {  pthread\_t t1, t2;  printf(“Printing partial sums of the array”);  pthread\_create(t1, NULL, sum, 0);  pthread\_create(t2, NULL, sum, 4);  pthread\_join(t1, NULL);  sum(8);  }  void sum(int startIndex) {  int partialSum = 0;  for (int i = startIndex; i < startIndex+4; i++) {  partialSum += matrix[i];  }  printf(“For index %d to %d = %d”, startIndex, startIndex+4, partialSum);  pthread\_exit(0);  } | | [4]  [4] |
| 4.  **CO3** | Consider the following processes with arrival time and burst time at a specific moment in the ready queue that needs to be scheduled.   |  |  |  | | --- | --- | --- | | Process | Arrival Time | Burst Time | | P1 | 2 | 8 | | P2 | 7 | 3 | | P3 | 5 | 10 | | P4 | 5 | 6 |  1. Apply Shortest Remaining Time First (SRTF) scheduling algorithm and show the following -    1. Gantt Chart    2. Average Waiting Time & Average Turnaround Time    3. Number of Context Switching 2. Apply Round Robin (RR) scheduling algorithm with quantum = 3 and show the following -    1. Gantt Chart    2. Average Waiting Time & Average Turnaround Time    3. Number of Context Switching 3. Find the best-suited algorithm between these two and give your reasoning. | [2+2+1]  [2+2+1]  [2] |