CL218 Data Structures Lab

Tuesday, June 23, 2020

Course Instructor

Ch. Usman Ghous

Serial No:

Final Term Exam
Spring Semester 2020
Max Time:3 Hour
Max Marks: 100
Exam Weight (Out of 100). 15

Roll No _____ Section _____

Guidelines for Submission:

1. You should submit only one PDF document and **all text should be typed**. Equations, f igures can be taken as pictures (all figures/equations can be pasted as images inside th at document).
2. You must submit your solution before due time via **Slate/Google Classroom**. Submis sions submitted after the due time shall not be considered.
3. If you don't finish every part of a question, don't worry! You can still submit what you've done to get marks based on your efforts.
4. In case of copied or plagiarized solutions in exam Or If a student provided help to ano ther student during exam both will be awarded "F" grade and it will affect the student CGPA.
5. Viva of any student can be conducted by the instructor after conducting an online exa m in case of any doubt.
6. This document should be submitted through LMS (**Slate/Google Classroom)**. But in worst case, you can email it within the deadline.
7. The file you submit should have the naming convention RollNo_DS_Final.

Question 1	Question 2	Question 3	Total
50	30	20	100

Question No. 1 **(60 Marks)**

Let us study Computer Science from the beginning.
We know that a primitive definition of a computer is a device which can take an input, process that input and turn it into a meaningful output.
Let us now consider a modern computer that can have multiple processes running and the operating system running on that system has to schedule all of them.
Your assignment starts from here. The table that follows describes the information you will need to attempt this assignment.

Sr. No.	Term	Description
1	Burst Time	Amount of time required by a process for execution. Also known as execution time or running time.
2	Waiting Time	Total time spent by the process in the ready state waiting for CPU.
3	Average Waiting Time	(Total waiting time for all processes / Total number of processes)
4	Turn Around Time	For a process (waiting time + burst time)
5	Average Turn Around Time	(Total turnaround time for all processes / Total number of processes)

Portion 1:

A very basic approach to schedule the processes would be to serve the processes according to their arrival, namely **First Come First Serve (FCFS)**. We will now have a look at **FCFS** and how it works in a computer system.
Suppose you have got n processes, and you have to find average waiting time and average turnaround time using **FCFS**. In **FCFS** processes are **queued** in the order that they have arrived and scheduled accordingly. Then process that arrives first will be executed first and next process starts only after the previous gets completely executed.

Process	Duration	Order of Arrival	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt chart:

Waiting time (P1): 0
Waiting time (P2): 24
Waiting time (P3): 27
Average waiting time: (0 + 24 + 27) / 3 = 17

Input & Process:
Suppose you have a file containing some processes (attached to the assignment file) containing process name, duration, order of arrival and arrival time. Access the file, change the process names to your last name e.g. P1 will be changed to Ghous1, P2 into Ghous2 etc. The rest of the information remains unchanged.
After you are done with it, load the file into your program and into a data structure of your choice ARRAY or LINKED LIST.
Implement the functionality of **FCFS** step by step using modular approach (Functions) and structures/ classes like the way we have done throughout the semester.
Hint: Use Linked List implementation of queues

Output:
You are expected to show the average waiting time and average turnaround time of the processes in the file and show it on console.

Points to Observe:

1. **This approach is non-preemptive. (The processes are not interrupted and they yield the resources themselves).**
2. **Average Waiting Time is not optimal (Could have been better if some other approach was used).**
3. **Processes cannot be serviced in parallel (Important). This results in "Convoy effect". Suppose there is a process with large burst time ahead in the ready queue and there are several processes with relatively smaller burst time behind this process. The scheduling algorithm is non-preemptive so the larger process will not be interrupted and hence a convoy effect will be in place.**

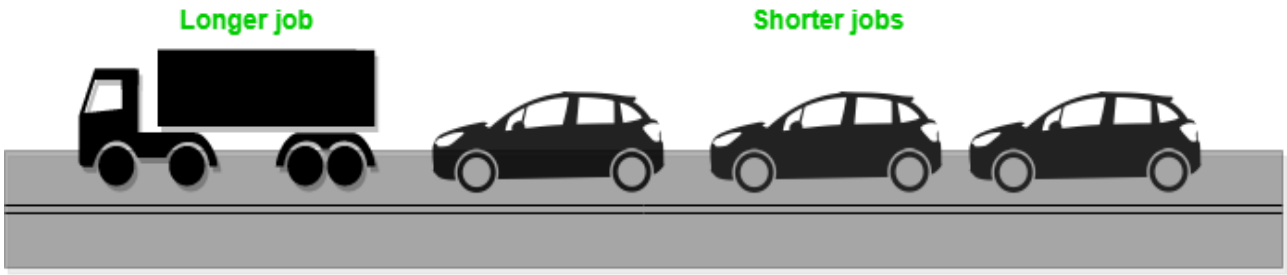


Figure - The Convey Effect, Visualized

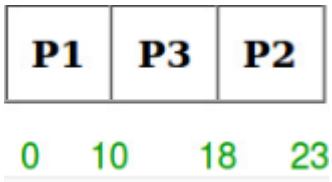
Portion 2:

Before we proceed on convoy effect, let us consider another scheduling algorithm. We have studied priority queue which is the hint for this part of the assignment. **Priority scheduling** is primitive to batch systems where each process is assigned a priority and process with highest priority is serviced first.

Processes having the same priority are served on FCFS basis. Now the situation will appear like this. Please note that for this portion we will assume that the numeric high in terms of priority is the higher priority i.e. priority 3 would be considered a higher priority than priority 1.

Process Id.	Burst Time	Priority
P1	10	2
P2	5	0
P3	8	1

Scheduled Queue:



Input & Process:

You have to access the text file named "priority scheduling" which contains data in the form of process name, burst time and priority. Again make changes to the process name as mentioned in portion 1. Load the file into an appropriate data structure and then operate it in the form of a priority queue. Also visualize it on the console step by step.

Implement the functionality of **Priority Queue** step by step using modular approach (Functions) and structures/ classes like the way we have done throughout the semester.
Hint: Use Linked List implementation of queues

Output:

You are expected to show the average waiting time and average turnaround time of the processes in the file and show it on console.

Points to Observe:

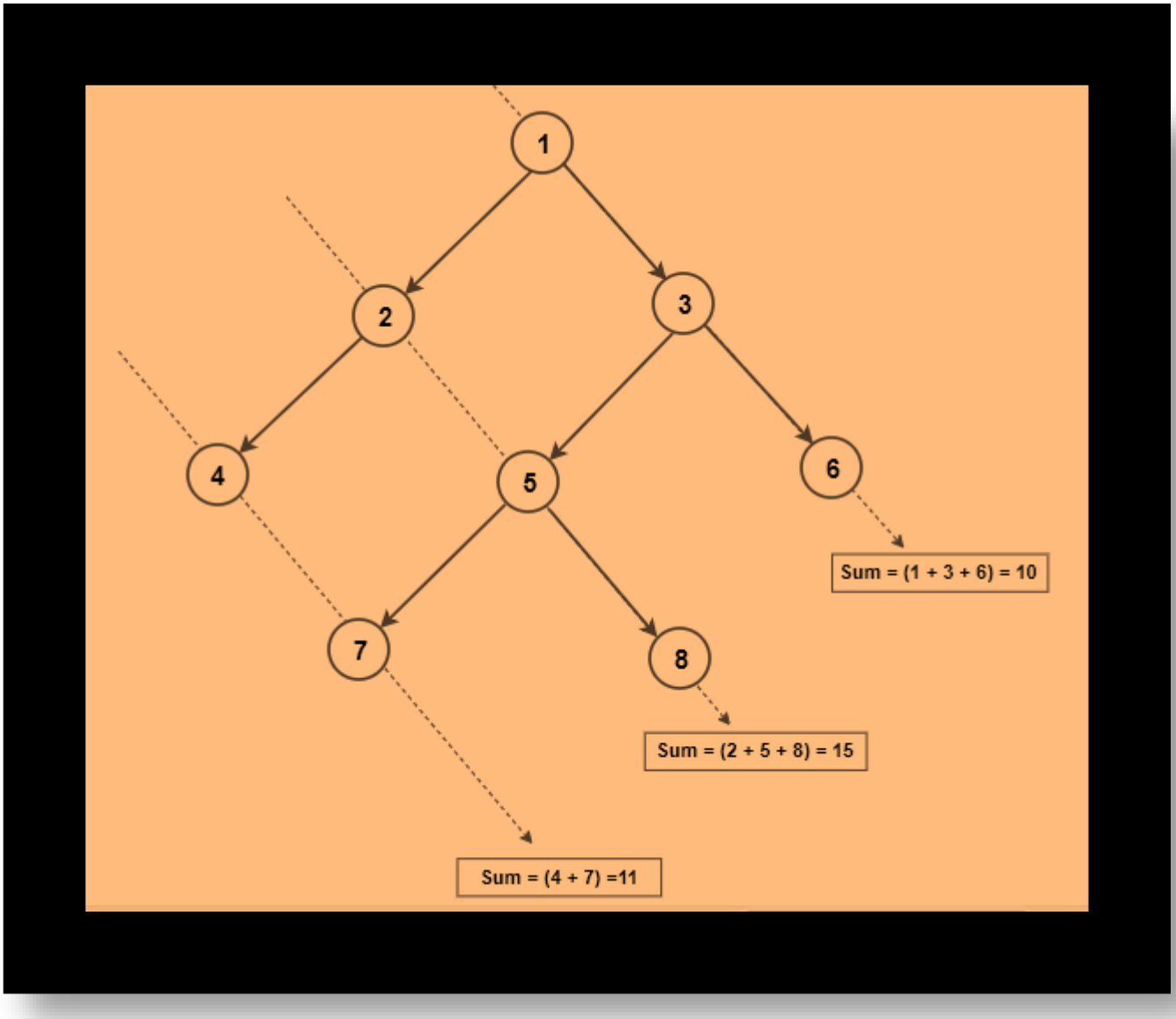
1. **This approach is non-preemptive. (The processes are not interrupted and they yield the resources themselves).**
2. **Average Waiting Time is not optimal (Could have been better if some other approach was used).**
3. **A problem with priority scheduling is starvation or blocking. Processes with very low priority may never get ahead in the queue and will starve for the CPU. A solution for that is aging which means gradually increasing the priority of process after some time so that processes don't starve.**

Portion 3:

Implement priority queue but instead of storing the priority of the process in the same data structure that you used before, store the priority in a MIN-HEAP. For processing the priority queue, extract the minimum value from the MIN_HEAP and then service the priority queue.

Question No. 2 **(20 marks)**

You have been given a binary tree as follows.



If you have a look at the tree you will see that there are nodes which occur as a slope…

1->3->6

2->5->8

4->7

Your task is to solve the problem of finding the sum of these slopes in a binary tree but with the help of hashing.

Hint:

1. Formulate an empty hash map.
2. Each key in the map represent a diagonal of the tree and its value is basically the sum of all the values occurring in that slope.
3. Consider a traversal algorithm of your own choice to update the map.

Question No. 3 (20 marks)

Mr. 'Y' has been given a graph and asked if it is tree. Help Mr. Y write a C++ program that takes a graph (representation of your own choice) and see if it is tree.