Question 1 A:

Lottery Scheduling is type of process scheduling, somewhat different from other Scheduling. Processes are scheduled in a random manner. Lottery scheduling can be **preemptive or non-preemptive**. It also solves the problem of starvation. Giving each process at least one lottery ticket guarantees that it has non-zero probability of being selected at each scheduling operation.
In this scheduling every process have some tickets and scheduler picks a random ticket and process having that ticket is the winner and it is executed for a time slice and then another ticket is picked by the scheduler. These tickets represent the share of processes. A process having a higher number of tickets give it more chance to get chosen for execution.

How it works ?

The lottery scheduling algorithm has a very different approach to scheduling processes. It works exactly as it sounds, each process in the ready state queue is given some lottery tickets, and then the algorithm picks a winner based on a randomly generated number. The winning process is awarded a time slice, in which the process may be complete or the time slice may expire before the process completes. The winning process goes back into the ready queue, receives its tickets for the next lottery, and the cycle continues.

Although each process will receive at least one (1) ticket per lottery, processes that are expected to be shorter will receive more than one ticket, thereby increasing the chance they'll be picked. In this way, the lottery scheduler resembles SJF/SRTF, in that it gives priority to what it thinks will be shorter processes. However, it is more fair that SJF/SRTF, because it does give each process a chance (albeit a small chance) to be selected for a time slice. Starvation is therefore avoided in the lottery scheduler.

The lottery scheduler degrades gracefully as the number of processes increases. Adding or deleting processes affect all other proportionally, independent of the number of tickets the process has.

A major drawback of the lottery scheduling algorithm is that it is inherently unpredictable. Due to the random nature of the scheduler, there is no way for OS designers to know for sure what will happen in lottery scheduling, possibly producing undesirable results. It is possible, for example, that one particular CPU-bound process could receive several time slices in a row, thereby allowing other I/O processes to languish without CPU time. For this reason, lottery

scheduling, while provably viable, is not used much in modern OS design.

Example – If we have two processes A and B having 60 and 40 tickets respectively out of total 100 tickets. CPU share of A is 60% and that of B is 40%.These shares are calculated probabilistically and not deterministically.

Explanation –

1. We have two processes A and B. A has 60 tickets (ticket number 1 to 60) and B have 40 tickets (ticket no. 61 to 100).
2. Scheduler picks a random number from 1 to 100. If the picked no. is from 1 to 60 then A is executed otherwise B is executed.
3. An example of 10 tickets picked by Scheduler may look like this –
4. `Ticket number -  73 82 23 45 32 87 49 39 12 09.`

   `Resulting Schedule -  B  B  A  A  A  B  A  A  A  A.`

   A is executed 7 times and B is executed 3 times. As you can see that A takes 70% of CPU and B takes 30% which is not the same as what we need as we need A to have 60% of CPU and B should have 40% of CPU.This happens because shares are calculated probabilistically but in a long run(i.e when no. of tickets picked is more than 100 or 1000) we can achieve a share percentage of approx. 60 and 40 for A and B respectively.

Question 1 B:

Earliest Deadline First (EDF) is an optimal dynamic priority scheduling algorithm used in real-time systems.
It can be used for both static and dynamic real-time scheduling.
EDF uses priorities to the jobs for scheduling. It assigns priorities to the task according to the absolute deadline. The task whose deadline is closest gets the highest priority. The priorities are assigned and changed in a dynamic fashion. EDF is very efficient as compared to other scheduling algorithms in real-time systems. It can make the CPU utilization to about 100% while still guaranteeing the deadlines of all the tasks.

EDF includes the kernel overload. In EDF, if the CPU usage is less than 100%, then it means that all the tasks have met the deadline. EDF finds an optimal feasible schedule. The feasible schedule is one in which all the tasks in the system are executed within the deadline. If EDF is not able to find a feasible schedule for all the tasks in the real-time system, then it means that no other task scheduling algorithms in

real-time systems can give a feasible schedule. All the tasks which are ready for execution should announce their deadline to EDF when the task becomes runnable.

EDF scheduling algorithm does not need the tasks or processes to be periodic and also the tasks or processes require a fixed CPU burst time. In EDF, any executing task can be preempted if any other periodic instance with an earlier deadline is ready for execution and becomes active. Preemption is allowed in the Earliest Deadline First scheduling algorithm.

Advantages :

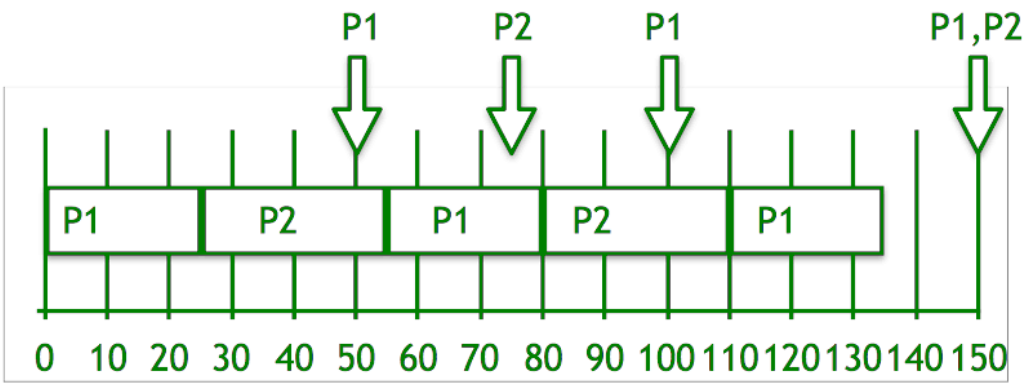
1. No need to define priorities offline.
2. It has less context switching than rate monotonic.
3. It utilize the processor maximum up to 100% utilization factor as compared to rate monotonic.

Disadvantages:

1. It is less predictable. Because response time of tasks are variable and response time of tasks are constant in case of rate monotonic or fixed priority algorithm.
2. EDF provided less control over the execution.
3. It has high overheads.

Example:

Consider two processes P1 and P2.
Let the period of P1 be $p_1 = 50$
Let the processing time of P1 be $t_1 = 25$
Let the period of P2 be $period_2 = 75$
Let the processing time of P2 be $t_2 = 3$



Steps for solution:

1. Deadline pf P1 is earlier, so priority of P1>P2.
2. Initially P1 runs and completes its execution of 25 time.
3. After 25 times, P2 starts to execute until 50 times, when P1 is able to execute.
4. Now, comparing the deadline of (P1, P2) = (100, 75), P2 continues to execute.
5. P2 completes its processing at time 55.
6. P1 starts to execute until time 75, when P2 is able to execute.
7. Now, again comparing the deadline of (P1, P2) = (100, 150), P1 continues to execute.
8. Repeat the above steps…
9. Finally at time 150, both P1 and P2 have the same deadline, so P2 will continue to execute till its processing time after which P1 starts to execute.

Question 1 C:

Mobile Operating Systems An OS is a software interface that is responsible for managing and operating hardware units and assisting the user to use those units. For mobile phones, OSs have been developed to enable users to use phones in much the same way as personal computers were used 1 or 2 decades ago. The most well-known mobile OSs are Android, iOS, Windows phone OS, and Symbian. The market share ratios of those OSs are Android 47.51%, iOS 41.97%, Symbian 3.31%, and Windows phone OS 2.57%. There are some other mobile OSs that are less used (BlackBerry, Samsung, etc.) [46]. In the next section, we will briefly explain each of these OSs.

I will talk about iOS Apple iOS is a closed-source code mobile phone OS developed by Apple in 2007; it is used by Apple-only products (iPhone, iPod, and iPad). The iOS architecture is based on three layers incorporated with each other. Cocoa touch is a layer that provides some basic infrastructure used by applications. The second layer is the media layer, which provides audio services, animation video, image formats, and documents in addition to providing two-dimensional (2D) and 3D drawings and audio and video support. The third layer is the core OS, which provides core services such as low-level data types, start-up services, network connection, and access .

I will choice two apps that need to work the access from the user like app maps and other apps (like WhatsApp, Facebook , Instagram all of this apps Operate and have access to the camera microphone …) that need to work the location services that uses GPS , Bluetooth , and crowd-sourced wi-fi hotspot and cell tower locations to determine your approximate

location , and this Vulnerability in the operating system and may compromise your security if someone breaks into your phone , because this I thought that this apps must have higher priority than regular applications

Question 3:

התוצאות הדפסתי אותם לקובץ primes-output

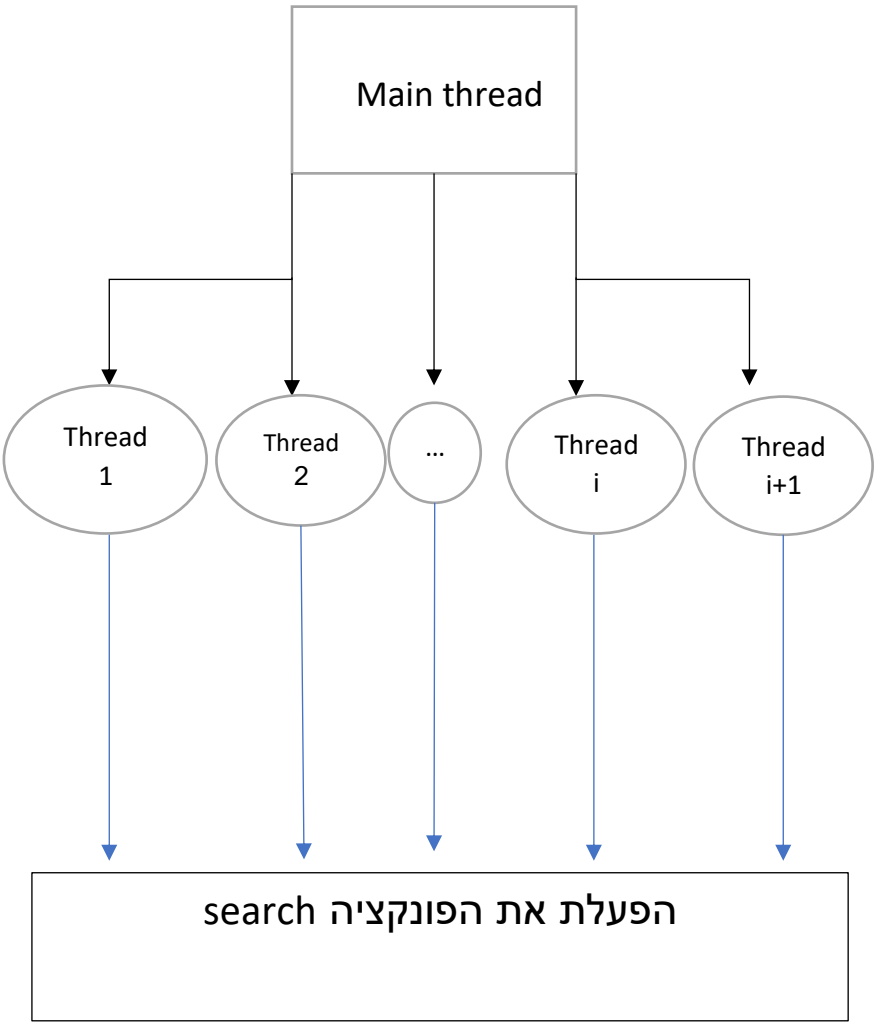
Question 4:

אלגוריתם :

אחרי קבלת המשתנים (nThreads, delta …) בהתחלה שמרנו את הטקסט במשתנה שהוא בעצם מכיל את כל האותיות בטקסט בנוסף ל n\ אחר כך בנינו מערך של המיקום של כל האותיות שהן האות הראשונה במחרוזת שאנחנו מחפשים שהפעולה הזאת ממומשת בפונקציה.

אחר כך בחרנו לחלק לטרדים כך שמחלקים את המערך של האינדקסים שבנינו על הטרדם כך שהטרד לוקח את המיקום הזה והולך לשם ומתחיל לחפש על המחרוזת שרצינו לחפש לפי הדלתה הנתונה (שזאת היא קריאה נוספת לפונקציה שמימשנו שהיא מתקדמת כל עוד מוצאת אות נכונה ומחזירה ערך בוליאני ולפי הערך הזה מחליטים אם נדפיס תוצאה או לא

דיאגראמה :


Main thread
Thread 1
Thread 2
...
Thread i
Thread i+1
הפעלת את הפונקציה search