Data Structures and Algorithms
Project Phase 1 Report

Number of Members: 3

Members' Info:

Member Name (EN)	Member Name (AR)	ID	Email
Amir Anwar	امير انور بخيت	9220166	Amir.Awd03@eng-st.cu.edu.eg
Akram Hany	اكرم هاني كرم	9220158	Akram.sallam03@eng-st.cu.edu.eg
Ahmed Hamed	احمد حامد جابر	9220027	Ahmed.Hamed03@eng-st.cu.edu.eg

Selected Data Structures:

List Name	Chosen DS	Justification

NEW List	Queue	• (FIFO order): The queue ensures that the processes are scheduled in the order they arrive, (As they are sorted by arrival time in the input file)
		• Fast Insertion and Removal
		  ○ inserting a new process in the queue is O(1)
		  ○ removing a process from the front of the queue is O(1)
		Therefore, using a queue to store new processes ensures that the process scheduler can efficiently manage incoming processes and schedule them for execution in an orderly manner.

TRM List	Queue	• terminated processes can be easily added to the end of the queue as they are terminated, and later be processed in the order they were terminated, ensuring that they are printed in ascending order by termination time.
		• Fast Insertion and Removal
		  ○ inserting a new process in the queue is O(1)
		  ○ removing a process from the front of the queue is O(1)
		
		A queue ensures that the processes are printed in the order they were terminated, which is important for bookkeeping and record-keeping purposes. This allows the process scheduler to easily track and manage the terminated processes and ensure that they are processed and outputted in the correct order.

BLK List	Queue	• (FIFO order): Processes are sent to IO in order of request (first in BLK list served first)
		• Fast Insertion and Removal
		  ○ inserting a new process in the queue is O (1)
		  ○ removing a process from the front is O (1)

FCFS RDY list	FCFSList	• Inherited from the ADT List, (Approved by TA. Eman) to allow for a function that searches the list and removes by ID for kill signal functionality>
		• Allows the flexibility to access elements by position when dealing with migration and/or work stealing.
		• Can be used in a FIFO manner
		• Fast insertion and removal
		  ○ Removal from the start: O(1) (used most of the time)
		  ○ Removal from the middle: O(N)
		  ○ Insertion at the end: O(1) (used most of the time)
		  ○ Insertion at the middle: O(N)

SJF RDY list	Priority Queue	• Prioritization: In the SJF processor, the process with the shortest remaining time is given the highest priority ensuring that the process with the shortest RT is always at front of the queue.
		• Efficient insertion and removal (using Minheap)
		  ○ inserting a new process is O (log n)
		  ○ removing a process from the front is O (log n)

List Name	Chosen DS	Justification

RR RDY list	Queue	• Efficient insertion and removal: Queues are efficient in inserting and removing elements, making them an ideal choice for storing processes in RR processors. Since processes are constantly being added and removed from the queue based on their time slice
		  o inserting a new process in the queue is O (1)
		  o removing a process from the front is O (1)
		• First-In-First-Out (FIFO) ordering: In an RR processor scheduling algorithm, processes are executed in the order in which they arrive in the ready queue

Processors List	List	• Implemented by using arrays to enhance the performance in this use case.
		  o Accessing an item: O(1)
		  o Inserting at the end: O(1) – (Requires no shifting and it is the only insertion used with processors.
		• List provides flexible position-oriented ADT that can store elements in a user-defined sequence. We can store all the processors in a single list, allowing us to easily iterate through them and schedule tasks to the appropriate processor based on the type of the processor

SIGKILL List	Queue<Pair>	• (FIFO order): kill signals are given sorted by Kill time in the Input file and processed in the same order.
		• The list will be stored and managed by the scheduler because it has access to all processors and the ID given is not guaranteed to be in a certain processor.
		• Fast Insertion and Removal
		  o inserting a new process in the queue is O (1)
		  o removing a process from the front is O (1)

IO (IO_R, IO_D) List	Queue<Pair>	• (FIFO order): IO requests come sorted by request time and processed in the same order.
		• Each process stores its list of IO requests.
		• Fast Insertion and Removal
		  o inserting a new process in the queue is O (1)
		  o removing a process from the front is O (1)