Data structures and Algorithms

Project phase1 report

Team name: 8-11_T13 **Number of members:** 4

Team email: m7mdrefaat550@gmail.com

Members' Info:

Member name	ID	Email
Mohamed Hazem Hamed	4220137	mohamed.abdelmaksood021@eng-st.cu.edu.eg
Mohamed Maher Refaat	4220140	Mohamed.refaat031@eng-st.cu.edu.eg
Mahmoud Ali Abbas	4220141	Mahmoud.Hassan031@eng-st.cu.edu.eg
Ghada Tarek Elboghdady	4220132	ghada.elboghdady041@eng-st.cu.edu.eg

Selected data structures:

List name	Chosen DS	Justification
NEW list	Linked queue	**Why:** It requires the First In First Out behavior, and has unspecified size **Operations and complexity:** Insertion -> O(1) Popping -> O(1)
RDY list (FCFS)	List using linked list implementation	**Why:** Because in the FCFS processor the order is important, and the linked list is dynamic data structure, resizable at run-time and efficient insertion and deletion. **Operations and complexity:** Insert a process = O(n) Remove a process = O(1) Kill a process = O(n)
RDY list (SJF)	Linked priority queue	**Why:** Because in the SJF processor it is important to insert processes in sorted

		way according to their CT and The element with the lowest CT is dequeued first and that what priority queue make. **Operations and complexity:** Insert a process = O(n) Remove a process = O(1) <u>No shifting</u>
RDY list (RR)	Linked queue	**Why:** Because in RR it follows FCFS algorithm but every process spend certain time (time slice) ,so the order is important and when I insert process it must wait the previous processes that inserted first to run first and when I dequeue I dequeue the first element inserted and that what the queue do **Operations and complexity:** Insert a process =O(1) Remove a process = O(1)
BLK list	Linked queue	**Why:** It requires First In First Out behavior with no known size. As a result Linked Queue is the right choice. **Operations and complexity:** Insertion -> O(1) Popping -> O(1)
TRM list	Linked queue	**Why:** It requires the First In First Out behavior, and has unspecified size **Operations and complexity:** Insertion -> O(1) Deletion -> O(1)
ListofProcessors	Dynamic Array	**Why:** No Popping or Inserting will occur to this DS every timestep; So minimum time for searching is important. Also indexing can be used which has constant time. **Operations & Complexity:** Searching -> O(n) Indexing -> O(1)
RunningProcesses	Dynamic Array	**Why:** As Insertion and Deletion are done randomly so No queue or Stack could be used. An array is a proper choice. **Operations & Complexity:**

		Searching -> O(n) Insertion-> O(1) Deletion->O(n)