**Cairo university, faculty of engineering Spring 20201**

**Computer Engineering Department**

# Data Structures and Algorithms CMPN102

# Data Structures and Algorithms Project Phase1 Report

#### Team Name: Team 14 Number of members: 4

#### Team Email: Kirollos.Ghoprial00@eng-st.cu.edu.eg

**Members’ Info:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Member Name** | **ID** | **Email** |  |
|  | **Khaled Hisham Sayed Taha** | **B.N: 21**  **Sec: 1** | **kha.hesham@gmail.com** |  |
|  | **Abd El-Rahman Noaman Lokman Ibrahim** | **B.N: 1**  **Sec: 2** | **bodynoaman1996@gmail.com** |  |
|  | **Abd El-Aziz Salah Mohammed Abdo** | **B.N: 2**  **Sec: 2** | **Abdelaziz132001@gmail.com** |  |
|  | **Kirollos Samy Hakim Ghoprial** | **B.N: 11**  **Sec: 2** | **Kirollos.Ghoprial00@eng-st.cu.edu.eg** |  |

|  |  |  |
| --- | --- | --- |
| **List Name** | **Chosen DS** | **Justification** |
| * Emergency missions | (Priority queue) | Operations:   * Enqueue * Dequeue * Peek   Complexity:  • O(1) → all operations  Except enqueue O(n)  Reason:  → Because the missions have a priority equation  so, the missions with higher priority must be executed first. |
| * Polar Missions | (Queue) | Operations:   * Enqueue * Dequeue * Peek   Complexity:  • O(1) → all operations  Reason:  Polar missions follow the concept of first come, first served and it does not depend on priority |
| * Mountainous missions | (Linked List) | Operations:   * Insert(from end)🡪 O(n) that’s not the best but we have to it’s a tradeoff after all * Remove🡪 O(1) (linked nodes implementation) * Search🡪 O(n)   Reason:  Although the first approach to come to mind is queue but there are some specific operations that would be done on this list cancellation and promotion require traversal across the list and removing from the middle which typically handled using linked list |
| * In Execution polar missions * In Execution Mountainous missions * In Execution Emergency missions | (Priority queue) | Operations:   * (enqueue) * (dequeue) * Peek   Complexity:  O(1)🡪 (all operations)  Except enqueue O(n)  Reason:  The execution time of a mission doesn’t depend only on arrival precedence but also on the execution time of each mission, missions that would be finished first should at front (the priority here should be calculated according to the remaining execution time for each mission  A separate list for each type is chosen for order of printing |
| * Checkup emergency Rovers * Checkup Mountainous Rovers * Checkup Polar Rovers | (Queue) | Operations:   * Enqueue * Dequeue * Peek   Complexity:  • O(1) → all operations  Reason:  Because all the rovers have the same checkup duration  so, the First to be enqueued should be the first to be done. |
| * Completed Emergency * Completed Polar * Completed Mountainous | (Queue) | * Operations:  - Enqueue  - Dequeue * Complexity  - O(1) -> All Operations   Reasons to choose this DS  - To make it easier while printing, first completed mission should be first printed |
| * **In mission rovers** | **(priority Queue)** | Operations:   * (enqueue) * (dequeue) * Peek   Complexity:  O(1)🡪 (all operations)  Except enqueue O(n)  Reasons:  It depends mainly on two factors the rover time to get to mission and the mission execution time so the rovers that would finish their mission first should be in front  We chose not to make a separate list as the printing problems was handled in (in Execution missions) so this list is just a transit for rovers |
| * **Emergency Rovers** * **Mountainous Rovers** * **Polar Rovers** | **(priority Queue)** | Operations:   * (enqueue) * (dequeue) * Peek   Complexity:  O(1)🡪 (all operations)  Except enqueue O(n)  Reason why this DS:  ->To access the fastest Rover to be enqueued first(Bonus)  If done without the bonus it would be a regular queue |
| **Events** | (Queue) | Operations:   * Enqueue * Dequeue   Complexity: O(1)🡪all operations  Reason why this DS:  ->First event is executed first then other events comes next in order. |

In all our Data structures we chose to use **Linked nodes** implementations for the following reasons:

* To avoid shifting in array implementations specially in queue (we don’t want to complicate things with circular array the linked implementation is straight forward)
* In addition, the priority queue is much easier and efficient when implemented by linked nodes
* In linked list DS the operations involve many removals, shifting so a linked implementation is a no brainer
* Although the size of most of the lists is predetermined, the complexity of operations led us to that choice