Robot Coordinate SRS - CS 3354.004 Ethan Fischer

Functions:

Function: Define new location or modify existing location on the map.
Description: The user will be able to use a command to create a new location on the map or update the name of an existing location on the map.
Inputs: A command such as '(a, 2, 5)' where 'a' is the name of the location (defined by the user, and '2, 5' are the x and y coordinates respectively.
Outputs: If a location at the coordinate (x,y) already exists on the map, the name of the location is modified, otherwise, a new location is defined at the specified coordinate on the map.

Function: Remove an existing location on the map.
Description: The user will also be able to remove a location already on the map.
Inputs: A command such as 'rm a' where 'rm' is the remove keyword and 'a' is the name of the location the user wants removed.
Outputs: If the given name is associated with a coordinate on the map, that location will be removed from the map. Otherwise, the system will inform the user that the location does not exist and therefore cannot be removed.

Function: Define a directed path between locations.
Description: The user will be able to control the direction from one location to another that the robots can travel along.
Inputs: A command such as '(a, d)' that indicates there is a path from location 'a' to location 'd' that the robot can travel across. If the path is a two-way path, the user will have to enter '(a, d)' and '(d, a)'.
Outputs: If both location names given are valid locations on the map, the system will "draw" a path from the first location to the second that the robots can travel on. If one or more of the locations does not already exist on the map, the system will show the user an error and prompt them to add that location at a user defined coordinate on the map.

Function: Remove a directed path from the map.
Description: The user will be able to remove a previously defined direction from one location to another to prevent the robots from taking that path.
Inputs: A command such as 'rm (a, d)' where 'rm' is the remove keyword and '(a,d)' is the path to be removed.
Outputs: If the path exists on the map, the system will remove that path and the robots will no longer be able to travel from 'a' to 'd'. Otherwise, the system will alert the user that the path does not exist and not modify the map.

Function: Create a new robot and put it on the map.
Description: The user will be able to add a robot to the map at a defined location and assign it a destination.
Inputs: A command such as '(r3,a,m)' where 'r3' is the name of the robot, 'a' is the starting location and 'm' is the destination.
Outputs: If the input is valid (there is no robot with the same name already on the map, the locations names given are associated with coordinates on the map) then the system will create that robot and place it at the starting location. If either of the inputs is invalid, the system will alert the user of the error and prompt them to either rename the robot, or define a new location with the input name.

Function: Remove a robot from the map.
Description: The user will be able to remove an existing robot if it is no longer in the area.
Inputs: A command such as 'rm r3' where 'rm' is the remove keyword and 'r3' is the name of the robot. **Outputs:** If the name given is associated with a robot on the map, the robot will be removed. If there is no robot on the map associated with that name, the system will alert the user of the error and remove nothing.

Function: Simulate the environment and keep track of the robot locations.
Description: The system will come with a simulation subsystem that will move the robots and keep track of their locations so that the information for each robot is available to the coordinate map system. **Inputs:** N/A
Outputs: The system will show the user a visual representation of the map, what locations are on the map and where each robot on the map is.

Functional Requirements:

1. The user will be able to type a command into the system to update an existing location or create a new location on the map. The system needs to validate user input so the user cannot create a new location at a coordinate outside the map boundaries or two locations with the same name.
2. The user will be able to type a command into the system to remove an existing location. The system needs to validate user input so that the user cannot remove a location that does not already exist on the map.
3. The user will be able to type a command into the system to define a directed path that the robots can travel on. The system needs to valide user input so that the user cannot draw a path to or from a location that does not already exist on the map.
4. The user will be able to type a command into the system to remove a directed path and prevent the robots from traveling on that path. The system needs to validate user input so that the user cannot remove a path that does not already exist on the map.
5. The user will be able to type a command into the system to create a new robot with a starting location and destination. The system needs to validate user input so that the user cannot create a robot that already exists, place a robot at location that does not already exist on the map, or send the robot to a location that does not already exist on the map.
6. The system needs to always be aware of each robot and their location on the map. When a new robot is created, the system will assign a path to the robot so that it can travel the shortest path to its destination without blocking any other robots on the map. For example, if a robot is currently moving from 'a' to 'd', the robot at 'a' cannot move to 'd'.

Non-Functional Requirements:

1. The system needs to pre-allocate the proper memory to the map so that all map modifications happen in-place which will be more time and memory efficient, improving system performance.
2. The system needs to be scalable to handle a large map with many locations and directed paths.
3. The system needs to be written in a portable language so that it can be easily run on a numerous types of devices (mobile, desktop, etc.).
4. The system needs to be implemented with authentication and security in mind so that a user's map data is secure and cannot be modified by people outside of the user's company.
5. The system needs to be well documented and easily maintainable so that future issues can be resolved quickly and so the system can adapt to future technologies.
6. The system needs to be resuable and flexible in the sense that different companies could use this system for their own unique needs. For example, a farm owner could use it to control robots harvesting crops on different parts of the property and bringing the crops back to a generalized location, and a factory owner could use it to control robots moving manufacturing materials to different locations in the factory.
7. The system needs to be built with reliable technology and backwards compatible with older versions so that the user never runs into any issues with running the system on their devices. The system needs to be built to continue working operations if an error occurs during an individual operation.