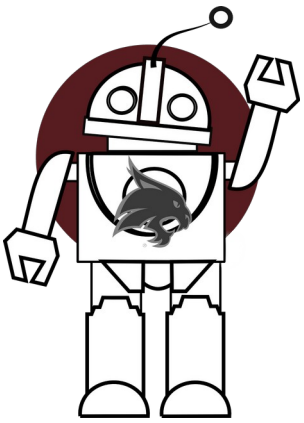
Bobcat Robotics

Software Requirements Document

Software Name: Robo Guider
Author: Gentry Atkinson
Version: 0.1.0



1. Introduction

Product Purpose: this product is being developed as a generalized robotic guidance platform. This product will take guidance input from the console and translate those values into positional data for a robotic platform.

2. Standards

Hardware: this product should be capable of running on a Linux platform with the gcc compiler and a means of text I/O.

Schedule: completion of this product is expected no later than 08 February.

Language: this project will be implemented using the C++ programming language.

3. System Description

System Context: this will be a self-contained piece of software that will not rely on external platforms for its basic operation.

User Characteristics: users will be trained robot operators who are proficient with computer operation. This project will not be responsible for user training.

4. Functional Requirements

Display: the user should be shown the current position of the robot (starting with 0,0) and a list of input options:

- **U**- move 1 space in the Y direction
- **D**- move -1 space in the Y direction
- **R**- move 1 space in the X direction
- **L**- move -1 space in the X direction
- **Q**- quit the program

The robots position should be correctly updated every time the user selects one commands. Every robot starts at position 0,0 and can move infinitely in any direction. The position should always be printed as **X, Y**.

You program should print an error message is the user enters a letter that is not in the menu.

Your program should print a short parting message when the user selects ‘Q’.

Interface: the display should be neatly formatted with the position and each command on a separate line.

5. Non-functional Requirements

• The program should be submitted as a .cpp file named [Author's LastName]_assignment0.cpp, with [Author's Last Name] replaced by the author's last name.
• The first three lines of the program should be the author's name, the date that the assignment was written, and a list of collaborators.
• The author is expected to follow the Style Guidelines.
• All text output should be neatly formatted.

5.1 Robot struct

The program should store the robots positional data in a **struct** with the following members:

- X- the current X value of the robot's position
- Y- the current Y value of the robot's position
- lastCommand- a character value representing the last move that the robot made

The initial values of the X and Y members should be set to 0. This could be done with a **constructor** if that's familiar or in the **main** function.

5.2 moveRobot function

The position of the robot **struct** should be updated using a void function called moveRobot. The parameters of this function are:

- Robot r: a Robot passed by reference
- char d: a character value representing one direction

The robots position should be updated as described in Section 4 (so if d == 'U', then increase r.Y by 1, etc.)

Only this function should be used to change the robots position after the initial position has been set.

6. Sample Test Cases

Input	Output	Stored in Robot struct
Start program	Robot’s position is 0,0 Please select: U- up D- down R- right L- left Q- quit	X: 0 Y: 0 lastCommand: anything
U	Robot’s position is 0,1 Please select: U- up D- down R- right L- left Q- quit	X: 0 Y: 1 lastCommand: ‘U’
L	Robot’s position is -1,1 Please select: U- up D- down R- right L- left Q- quit	X: -1 Y: 1 lastCommand: ‘L’
K	Invalid command	X: -1 Y: 1 lastCommand: ‘L’
Q	Goodbye	nothing