Program Description:

To start, I developed a basic program to detect barcodes with the use of image recognition and sensors. The optimal threshold value determined was 4600, consistently applied in the final code. The focus then shifted to movement. A pattern-based search using ninety-degree turns was chosen as it suited the relatively consistent course layout, with minor variations, simplifying implementation. The initial concept involved using ninety-degree turns to guide the robot from one barcode to the next due to the barcode configuration. This alignment made reading the barcodes effortless. In the simulation, a ninety-degree turn took 1850ms to perfect. However, in reality, the robot couldn't get close enough to the wall for the first barcode without collision.

Testing on the actual course revealed that by adjusting the threshold value upwards, the robot could approach the wall closely during the ninety-degree turn and align accurately with the barcode. However, drift of the robot was not anticipated. Our robot had a tendency to drift left, disrupting alignment and barcode reading. Adjusting the turn times after barcode reading became necessary. Following the first barcode reading, the turn time was modified, angling the robot slightly to the right for a curving return to barcode alignment. Similar adjustments were made to target the third barcode. Each turn corresponded to a decision point, incrementing the decision variable. After each turn, the robot moved forward using a timer event.

With this resolved, I proceeded to develop identification subroutines for barcode reading. Three separate subroutines were created. The first identified whether the barcode indicated a right or left direction, adjusting the "R" or "L" variables accordingly. The second subroutine relied on the initial turn direction ("R" or "L") and the indicated direction from the barcode to determine the path, particularly noting a special case when the course configuration was "LL". The third subroutine reused code from Task 1 for barcode identification. Each identification subroutine included a light-up and beep for direction indication.

Managing hazard lines followed. A "hazard" variable incremented upon detecting and not detecting a black line. When "hazard" reached 5, the robot stopped for five seconds, changed color to red, beeped, and incremented the "decision" variable. These actions occurred when the decision variable equaled 4. Lastly, the treasure room was addressed. After hazard lines, the robot turned left or right based on the "path" variable determined from reading the second barcode. This variable also governed subsequent turns to reach the treasure room. When the decision variable reached 7, a 3700ms timer initiated during forward movement. Afterward, the robot turned into the treasure room. Incrementing the "stop" variable upon encountering gray, black, and gray colors, the robot stopped when "stop" reached 3. The displayed color matched the "color" variable.

In summary, the program's strengths were its simplicity and straightforwardness. However, weaknesses included the absence of a recovery state and failure to consider external factors like robot drift. Incorporating the line-follow program for barcode alignment could have been a more robust solution.