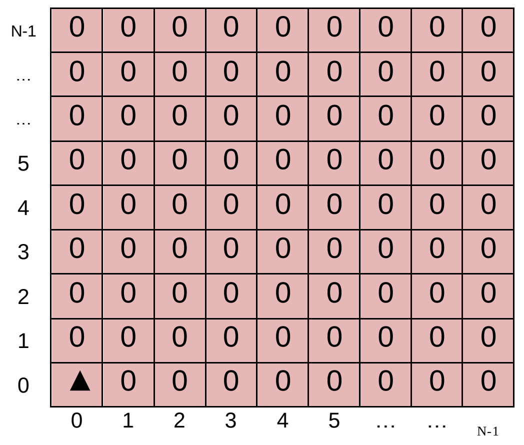
Concordia University
Department of Electrical and Computer Engineering
COEN 488 – Software Testing and Validation (Summer 2023)
Project (Lab Work): Robot Motion

The objective of the project is to test the practical use of the software testing techniques covered in the course. The project will be performed over supervised lab sessions. There will be deliverables (see the Project Milestones section), including a final demonstration of all the artifacts developed during the project. The demo will be performed during the lab sessions.

NOTE: The marks associated with each aspect of the project will be communicated to you by the TA. Also, the TA reserves the right to make some modifications to the deadline and deliverables depending on how the labs progress.

1. Project Description

Imagine a robot that walks around a room under the control of a Java program. The robot holds a pen in one of two positions, up or down. While the pen is down, the robot traces out shapes as it moves; while the pen is up, the robot moves about freely without writing anything. The room will be represented by an N by N array called floor that is initialized to zeros. Initially the robot is at position [0, 0] of the floor, its pen is up, and is facing north (as shown in the below figure).



The robot moves around the floor (i.e. the array) as it receives commands from the user. The set of robot commands your program must process are as follows:

Command	**Meaning**
[U|u]	Pen up
[D|d]	Pen down
[R|r]	Turn right
[L|l]	Turn left
[M s|m s]	Move forward s spaces (s is a non-negative integer)
[P|p]	Print the N by N array and display the indices

[C|c]	Print current position of the pen and whether it is up or down and its facing direction
[Q|q]	Stop the program
[I n|i n]	Initialize the system: The values of the array floor are zeros and the robot is back to [0, 0], pen up and facing north. n size of the array, an integer greater than zero

* By default, input of command [M s|m s] and [I n|i n] should follow the format of command character following by zero or one space and then an integer greater than zero.

** You can override this default requirement on input. Please note the new overriding requirements must be clearly specified in your project deliverables (please refer 3. project milestone task 1)

2. Project Teams

Projects will be executed by teams. Each team will **contain two - four members**. The project follows the agile development using the Scrum method. The scrum roles are rotated among the members according to the deliverables of the project along the timeline. The Moodle site provides a group selection tool. Please provide a name/title for your group. Undergraduates and graduates cannot form a joint project.

For those have difficulties in finding a group to join, please also using the good selection tool to put your individual name as one-member-only group. The TA will randomly assign groups.

At the end of the project, each team member will fill in the evaluation of the contribution of each of the team members to the project.

3. Project Milestones and Deliverables

In this project, one will proceed following an iterative and incremental process. One will be doing development and testing work.

- Task 1: Development and Verification (**Deadline: Week 2 – Friday 11:59pm**)

Write a Java program that simulates the robot capabilities to move through the floor. The program must keep track of the current position of the robot at all times and whether the pen is up or down. As the robot moves with the pen down, set the appropriate elements of the array floor to 1 (no matter how many time it has been traced). When the 6 command (i.e. print) is given, wherever there is a 1 in the array, display an asterisk; wherever there is a zero, display a blank.

Here is only a command line example of how the execution of the program should look like. The program can have a graphic user interface as well.

```
>Enter command: I 10
```

(This should initialize the system as a 10 x 10 array. The values of the array floor are all set to zeros and the robot is back to [0, 0], pen up and facing north.)

```
>Enter command: C
>Position: 0, 0 - Pen: up - Facing: north
```

(This is output by the program to indicate the position of the robot and the whether the pen is up or down)

```
> Enter command: D
```
(Now user orders the robot pen down)
```
> Enter command: C
> Position: 0, 0 – Pen: down - Facing: north
```

```
> Enter command: M 4
```
(The user orders the robot move 4 cells forward, the new position of the robot is now 0, 4)
```
> Enter command: C
> Position: 0, 4 – Pen: down – Facing: north
```

```
>Enter command: R
```
(The user orders the robot turn right)
```
> Enter command: C
Position: 0, 4 - Pen: down – Facing: east
```

```
>Enter command: M 3
```
(The user orders the robot move 3 cells forward)
```
> Enter command: C
Position: 3, 4 - Pen: down – Facing: east
```
(The new position is now [3, 4] since the robot turned right)

```
>Enter command: P
```
(The user wants to display the floor. You don’t need to print any background color of the array)

```
9
...
...
5
4   *   *   *   *
3   *
2   *
1   *
0   *
    0   1   2   3   4   5   …   …   9
```

```
>Enter command: Q
```
(End of program)

<u>Basic Requirements</u>

- Each development should adopt a Maven build file for importing / exporting a project that includes all the source code and tests. All the dependencies of testing packages should be setup within the Maven build file.
- Use a time tracking tool to record the development time vs the test case development time. An example tool is https://wakatime.com/

Deliverables and Milestones (Developer centric solution) – 10 marks:

- A working application (the project can be downloaded, compiled, and run in any Java platform)
- The source code is available in the github repository to share.
- Lab Demo to TA in Week 3 (Please book time with TA. If no slot available, please follow instructions from the TA to upload a recorded demo video)
- Unit test code should be included in the source code.
- A detailed report in PDF that includes
  - Github URL
  - Requirements. Each requirement is a one or two-line statement that describes a specific function of the system. Each requirement should have a unique identifier i.e. R1, R2.
  - Screen-shots of each function to fulfill the requirements.
  - A table mapping requirements and unit test cases.
  - Test cases execution results (fail or pass).
  - Tracked application development time vs testing case development time

Or Deliverables and Milestones (applied AI solution) – 10 marks:

- A working application (the project can be downloaded, compiled, and run in any Java platform)
- The source code is available in the github repository to share.
- Lab Demo to TA at Week 3 (Please book time with TA. If no slot available, please follow instructions from the TA to upload a recorded demo video)
- Unit test code should be included in the source code.
- A detailed report in PDF that includes
  - The code generation tool description and code generation method adopted
  - Conversion of the project requirements to tool inputs such as prompts and configuration
  - Github URL
  - Requirements. Each requirement is a one or two-line statement that describes a specific function of the system. Each requirement should have a unique identifier i.e. R1, R2.
  - Screen-shots of each function to fulfill the requirements.
  - A table mapping requirements and unit test cases.
  - Test cases execution results (fail or pass).
  - Tracked application development time vs testing case development time

- Task 2: Code Analysis and Software Release (**Deadline: Week 3 – Friday 11:59pm**)

This task aims to use Java code coverage tools to find what parts of the code are covered by tests and what parts are not. The results can help to decide if unexecuted parts should stay or remove. Choose a code analysis tool to produce the measurement for the following metrics:

a) Function coverage: how many of defined functions have been called;

b) Statement coverage: the number of statements that have been successfully executed in the program;
c) Path coverage: how many paths of the control structures in the program (if statements or loops) have been executed;
d) Condition coverage: how many Boolean expressions have been validated inside your program;
e) Line coverage: how many lines of the program have been executed.

<u>Deliverables and Milestones – 10 marks:</u>

- Demo to TA at Week 4 (Please book time with TA. If no slot available, please follow instructions from the TA to upload a recorded demo video)
- Updated Maven build file for new tools used;
- Update the task 1 report in PDF that includes two new sections for code analysis and software releasing.
    - Github link
    - Predefined the code coverage threshold values for releasing
    - Code coverage results for (a) to (e), with screenshots from the code analysis tool.
    - Discussion of the code coverage results in term of the threshold values and decision making for releasing.
- Submit the updated report in a single PDF file.

<u>Task 3: Black-box and White-box Testing (**Deadline: Week 5 – Friday 11:59pm**)</u>

Each team is performing the QA role. The assignment of QA-Dev mapping will be provided by Week 4 Monday. The github link for the application and the report developed by another team (Dev team). Each team will be required to test the application using various techniques including black-box and white-box testing techniques or any other technique appropriate. Apply white-box testing techniques as follows:

a) statement coverage (show percentage covered > 50%)
b) decision coverage (show percentage covered > 50%)
c) condition coverage (show percentage covered > 50%)
d) multiple condition coverage (show percentage covered > 50%)
e) Select one function and apply mutation testing
f) Select one function and apply data flow testing.

<u>Deliverables and Milestones – 10 marks:</u>

- Lab Demo to TA at Week 6. (Please book time with TA. If no slot available, please follow instructions from the TA to upload a recorded demo video)
- Define the Maven build file for tools for the test case development together with the original app source code.
- A QA test report in PDF that includes:
    - A github link of all the source code of the test cases
    - A list of test cases developed for (a) – (d)
    - A list of all the mutations generated by mutation testing tools (e)
    - A list of test cases developed for (f)
    - For each of the test cases, show the test result with screen-short.
    - Test conclusion. Combine all the above test case results to derive the test coverage for the application under test; and define the criterion to draw conclusion for the QA test result.

  - Summary and comments for the Dev team.

Task 4: QA Team’s Presentation and Dev Team’s Response (Week 6 Thursday Lecture Time) – 5 marks

- Selected QA team and Dev team present together at the lecture time. Volunteering teams are encouraged. A bonus point will be applied. Available space is limited and following the first-come-first-get rule. If a group is required to do demonstration by the lecturer, but do not participate without approval. The recorded video submission is not an option.
- Other teams (the QA and Dev teams jointly) that do not present should submit the presentation in a recorded video.
- **Submission of presentation slides (or with recorded video) by Week 6 Friday 11:59.**