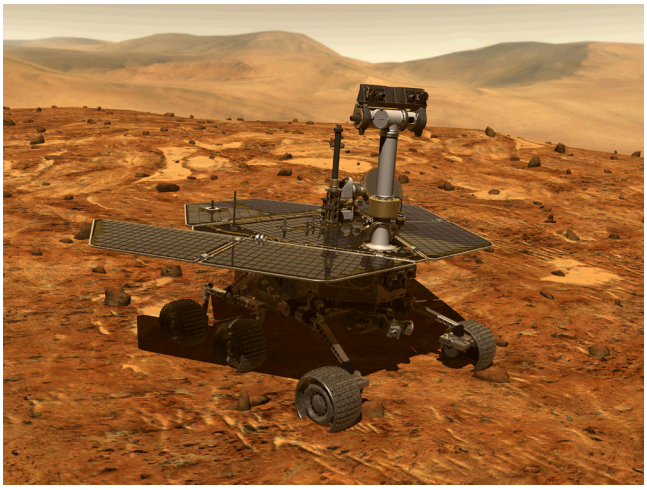
CSC 192 Assignment 2

Roving on Mars

... or ...

Earth Attacks!

Due: Oct. 6th @ 11:59 PM



In this assignment, you will design a decision-making module for a Mars land rover. The Mars rover has just landed and is now faced with obstacles. Your job is to navigate through the obstacles to reach the "mineral deposit point" so that the rover can collect samples for analysis to send back to Earth.

Luckily, the the rover has landed in a flat surface of Mars, but unfortunately there are many large rocks preventing passage. Your job is to find a path, *any path* (eg. it doesn't have to be the shortest path), from the rover's landing point to the mineral deposit points.

Your input files will contain some information about the landing site. The landing site will be represented as an r (rows) by c (columns) grid. Some areas will be passible (`.`) and some impassible (`#`). The rover's landing location is marked as an `L` and the mineral deposit point is marked by `X`.

An example input file looks something like this:

```
6
10
...#.....#
#.##.##.L#
#......#..
###.......
#X....#...
..#..##..#
```

The first line of input contains r, the number of rows in the landing site. The second line of the input contain c, the number of columns in the landing site. Then, the landing site is given, 1 line per row. Assume that you will always be given two integers, one-per-line, at the beginning; however, the landing site might be invalid (not represented in the format specified above, contains extra whitespace, or does not contain exactly 1 of each `L` and `X`). If it is, your program should print out "invalid input" (to `stderr`) and exit returning an errorlevel of 1 to the operating system. Note: since you don't know how large the landing sites will be ahead of time, you will have to use dynamic memory to store the landing site.

Your job, as the decision-making module, is to print out a path (sequence of directions) that lead the rover to the mineral deposit point. The directions must be one of: `north`, `east`, `south`, or `west`. The directions need to be printed from top-to-bottom, one per line. Any paths that lead the rover off the landing site or into rocks are dangerous and will be considered incorrect.

An example of a correct output given the example below would be:

```
south
south
west
west
west
west
south
west
west
west
```

Another example of correct output would be:

```
north
west
west
west
west
south
south
south
south
west
west
west
```

If there is no path then print

```
no path
```

and exit, returning an errorlevel of 0 to the operating system (the program succeeded in finding no path – this is not considered an error). Please make sure the output is formatted exactly as specified.

You may be tempted to use recursion for this assignment. Do not. You are not allowed to. You must write an iterative (eg. using loops) solution.

Advice on how to complete this assignment. Before jumping into the implementation, think about an algorithm that solves the problem. Convince yourself that the algorithm you come up with is correct. Often, time spent programming could be wasteful if you haven't thought about the problem enough before-hand.

A few example test files will be available from BlackBoard, both valid and invalid. Make sure you make a few test files yourself as well, to test cases not covered in the student test files. We will test with our own (hidden) test files as well, to make sure your decision-making module is robust to all situations.

Make sure that your assignment is well-documented and is written using good programming style, as specified in the assignment guidelines and style guidelines! Hand in everything except any object files and executables. As usual, your program has to compile without warnings and errors using:

```
gcc -Wall -o marsrover marsrover.c
```

on the SF1012 lab machines.