Assignment 1 - Oregon Trail
CSCE 4114 – Embedded Systems – Fall 2019
Due Date – Friday, October 4th, 11:59 PM CT

Description: You will develop a simple version of the "Oregon Trail" game from 1971. The assignment will develop file input/output, command line input/output, dynamic memory, structs, unions, and pointers.
Expected Operation: The program is to be written in C. The following will define the behavior of the program you will develop:

1. The program will open to a prompt that will allow a user to start the game, read out instructions, or exit
2. When played, the program will read and parse a file as the first argument passed to the executable that contains (1) an amount of "health" to be assigned to the character, and an NxN grid of characters that define what is in the map
3. The map will be defined as an NxN grid with pointers to cells above, below, to the right, and to the left
4. Each square will contain one of the following:
   (a) A river
   (b) A wild animal
   (c) A disease
   (d) A boundary (Ocean)
   (e) Nothing
5. After loading the map, the system will print out a viewport, which is a 3x3 ASCII representation of your current location, and "health"
6. The user's viewport will have unknown values for a potential disease, animal, or nothing until the user attempts to traverse into that area. Rivers and boundaries will show as their defined character
7. The user will then attempt to traverse from the bottom right to the top left of the map through inserting 'U', 'D', 'L', or 'R' into the prompt.
8. When a user attempts to traverse into a square, one of the following will happen:
   (a) A river crossing – The user will be prompted to determine if they want to attempt the cross. A confirmation of 'Y' will cross into the square, while 'N' will retreat back to the current square
   (b) A wild animal – The user will be asked if they want to hunt the animal (again with a 'Y' or 'N' confirmation prompt). The animal will provide health in the form of "food", but will also deal damage.

(c) A disease – The user will enter into the square with no confirmation prompt, and will immediately be dealt damage

(d) Nothing – The user will enter into the square with no confirmation prompt

(e) Boundary (Ocean) – The user will be told they may not move in that direction

9. If the user runs out of health, then the user should be prompted that they have lost the game, and will be returned back to the main screen

10. If the user makes it to the final location at the top right of the map, they should be prompted that they have "Won" and be returned to the main screen

Rubric: The project will be graded according to the following rubric:

Category	Description	Percentage
Pass Given Tests	There are a set of 2 maps with 3 tests for map 1, and 2 tests for map 2 provided on the website. Each test is worth 10% of the grade for this project.	50%
Pass Hidden Tests	I will run a set of 2 hidden tests (with a different map) on your executable to test operation. Each is worth 10% of the project grade.	20%
Coding Comments & Style	Use appropriate coding styles. Comment functions with a description about their behavior, any parameters, any return values, and any shared variables which it manipulates.	15%
Report	A simple, one- or two-page report. The report should have the project name (e.g. Oregon Trail), a short description of what you did, and the outcomes (e.g. Did it pass all example tests, if not, why not, etc...).	15%

Table 1: Grading Rubric

Given Tests: The known tests will be provided as files in the information section of the project on the website.

These documents should be uploaded through Blackboard to the TA before the beginning of the next lab. Documents turned in after this deadline are subject to the course late policy.

Implementation:

Map: The map will be read in from a file, which will be passed into the program using a command line argument to the location of the file. (To run the program, you will execute something like "./hw1 inputFile.txt") The map will be read into 2-dimensional linked list of nodes, from which you will be deriving your viewport. Any changes made to the map (by hunting an animal) will be updated to the map. Two sample input files have been provided that define the amount of initial health and the map, using the first letter of each danger as the marker (i.e. Grizzly is marked with G, Elk with E, River with R, etc.). The map is

surrounded by O's, which is for an impassable ocean space. Anything beyond the map is more ocean.
Viewport: Your viewport (3x3) can be thought of as a minimap, or what the player can currently see. All squares except ocean, river, your starting point, and the end point are initially marked as undiscovered. Your viewport should reflect this by displaying a 'U' in that square. You will print the viewport after each action. Discovered squares will display the threat that is in the square. The viewport will be a 3x3 2-dimensional linked list. The viewport will be constructed by copying each node from the original map into dynamic memory. Each node will only write back to the map whenever the player will be moving to where that node is no longer displayed. At that point, the node will copy its changed contents back to the map, and then free itself from memory.
Node: A node will be a struct object which contains the following variables:
Threat: A union which contains either a disease or an animal.
ThreatType: An enum which contains either "DISEASE", "ANIMAL", "NOTHING", "RIVER", or "OCEAN"
XLocation : An integer of the x location in the map
YLocation: An integer of the y location in the map
Discovered: A Boolean value as to whether or not the node has been discovered
Node *Up, *Down, *Left, *Right: 4 pointers to the nodes to the top, left, right, and below the node

Threat Types: The threat types are as follows:
Animals: Disappear after hunting

1. (G)rizzly – -10 health
2. (B)oar – -5 health
3. (H)are – +5 health
4. (E)lk – +10 health

Disease: Persist

1. (D)ysentery – -15 health
2. (F)lu – -10 health
3. (C)old – -5 health

River Crossing: -20 health per river square traveled.

Example Scenario:

```
Assume the following map for 50 health:
===================
||Z|N|N|N|N|N|N|N||
||R|R|N|N|N|N|N|N||
||E|R|E|N|B|N|H|N||
||D|R|R|E|E|H|H|H||
||D|N|R|R|R|R|R|H||
||H|G|N|N|E|E|R|R||
||F|N|N|G|H|G|C|E||
||E|N|F|D|N|G|G|X||
===================

When the user enters the game, they are presented with the following 3x3
map and health status:

Health: 50
=========
||U|U|O||
||U|X|O||
||O|O|O||
=========
This viewport says the user is surrounded above and to the left by unknown
squares, and to the right and down by impassable ocean. If the user chooses
to travel right or down, they will be prompted with "Cannot pass ocean tiles"

Assume the user chooses left, the user would then be prompted with:
"You have encountered a Grizzly bear, would you like to hunt (y/n)?
Assume the user chooses not to lose 15 health, and instead retreats.
The user is then prompted with the following viewport:

Health: 50
=========
||U|U|O||
||G|X|O||
||O|O|O||
=========

The user now chooses to travel up, encounters an Elk (10 food) and hunts.
The user moves into that square, and is presented with the following viewport:

Health: 60
=========
||R|R|O||
||U|X|O||
||G|N|O||
=========
```

The empty space left by the user is now a Nothing (N). The user at this point chooses to move left to avoid crossing the river, and runs into the common cold (-5) with some status message, e.g. "You have contracted the cold (-5 health)", and is prompted by the following viewport:

```
Health: 55
=========
||U|R|R||
||U|X|N||
||U|G|N||
=========
```

The user chooses to ford the river at this point, and travels up. They are promtpted with "Are you sure you wish to ford the river? (y/n)". A y will have the user enter the river, losing 20 health, and prompted with:

```
Health: 35
=========
||R|R|H||
||U|X|R||
||U|C|N||
=========
```

The user can choose to continue to ford the river for an additional 20 health, turn back and catch the cold again, or face the unknown.