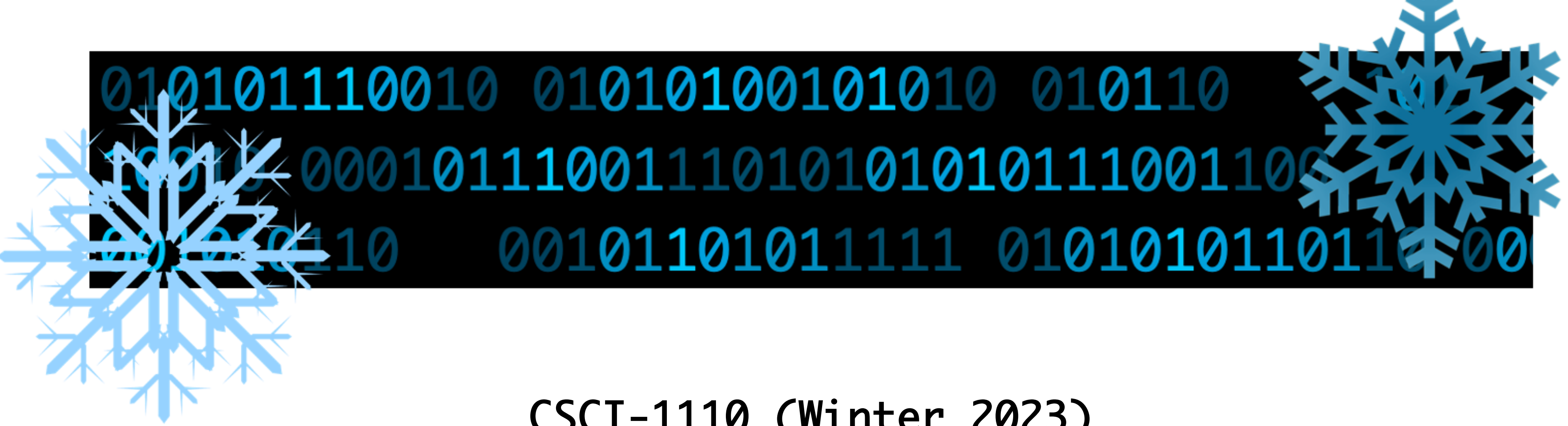


CSCI-1110 (Winter 2023)
ASSIGNMENT 4

The purpose of this assignment is to reinforce your understanding of object-oriented design and linear data structures such as lists and queues. For this problem you will be provided with sample tests in Codio. All input is done via the console and all output is to be done to the console as well. You must submit your assignment via Codio, and you can test it by submitting your code.

Note: for this assignment you are expected to install and use an IDE (other than Codio) to develop your code. Please see How-To videos in the How-To Tutorial Section of our Brightspace page on how to install an IDE.

As always, there will be points allocated based on code clarity and code quality. Check out the Code Style Guidelines in Brightspace.

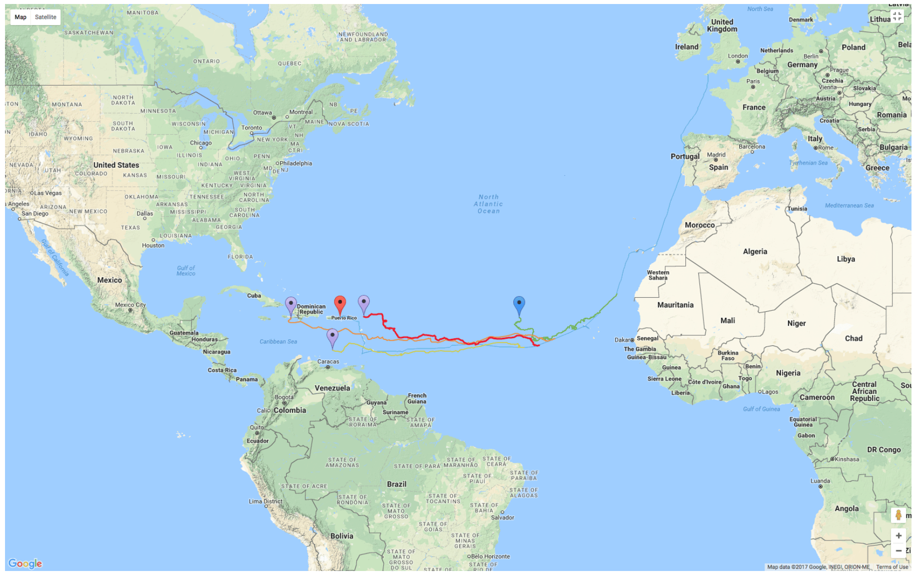
DUE DATE:

This Assignment is due on Monday, April 3rd at 14:00 (2PM) Halifax time.

IMPORTANT NOTE:

Before starting this assignment, make sure to read through all aspects of it. Make plan!

Modern Message in a Bottle



Dr. Siegel and Eliana prepare to deploy “Drifter E” (their modern message in a bottle) five days out from Cape Verde on the way to Barbados. The path of this buoy is given in red on the right. (https://www.cruisingworld.com/modern-message-in-bottle/)

For centuries, sailors and those near oceans have been sending messages across the seas by putting their message into a bottle. Their messages will either sink, end up in an ocean eddy or reach an unknown person in faraway lands. In the era of postal services and internet, we can sometimes learn where these messages end up. For instance, a message deployed by a fisherman off the coast of Nova Scotia was found on a beach in the Bahamas. (https://www.cbc.ca/news/canada/nova-scotia/nova-scotia-message-in-a-bottle-manjack-cay-bahamas-1.6761789)

The message in a bottle is as old as sailing itself, so why not learn more about where it goes when your bottle goes overboard? With ocean current models, we can predict where these bottles will end up (THAT will be our task!) and follow them to their destinations.

Model Behaviour

There will be several parts to this assignment that you will have to consider. You will be creating objects to model various aspects. Your first real step is to plan out how you will model each aspect of this problem.

The Earth's Surface

First, you will be given a map, depicting the Earth's surface. Some of the Earth's surface is made up of land (L) and some of ocean (O). Each cell will represent a portion of the Earth's surface. Your first input will be the height and width of the map of the Earth's surface you will be given. You should at first assume that all cells are ocean (after all, a large portion of the Earth's surface IS covered by ocean!). This will be followed by the locations (pairs) of each land cell.

The following input depicting a map of the Earth's surface will ultimately depict the land (green/L) and ocean (blue/O) as depicted to the right:

```
4 5
0,0 1,0 1,1 2,4 3,3 3,4
```

L	O	O	O	O
L	L	O	O	O
O	O	O	O	L
O	O	O	L	L

Surface Currents

Next, you will be given a map, depicting the direction of ocean currents (the direction that the surface ocean water is flowing). These ocean currents can be moving water north (N), south (S), east (E) or west (W). You will receive a current direction for each cell of your map. However, these details are not relevant over land. The map of the surface ocean currents to the right will be input as:

```
E E E S E S W S W W W E E E
```

🟩	E	E	E	S
🟩	🟩	E	S	W
S	W	W	W	🟩
E	E	E	🟩	🟩

Note that currents are not given for land cells. So you will want to check that property of your Earth's surface before applying a current.

Bottles

The most important part of this process is going to be keeping track of our bottle(s)! Each bottle has a message in it, and it will also have drift path. This is where your work with linear data structures will come in handy!

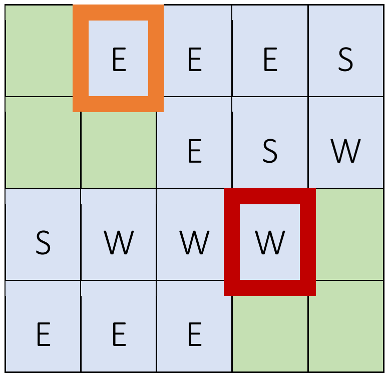
Your next line of input will tell you how many bottles to expect.

Bottle - Name, Location & Message

We are going to track where each bottle is located over time. As such, you'll want to have a way to store and consider that location. Typically, on a map, the north/south position is noted as latitude and the east/west position is denoted as longitude. For each buoy, as input, you'll receive two lines of information. The first will have the starting coordinates of the bottle on the grid followed by the bottle name. On the next line, you'll receive the message held within the bottle.

For instance, if you are given the input

```
    2
    0 1 Drifter E
    Smile often and “seas” the day!
    2 3 Floaty McFloatface
    Keep clam and carry on!
```



the bottle named "Drifter E" (which is carrying the message "Greetings from far away lands! Smile often and "seas" the day!") will start in the cell outlined below in orange and the bottle named "Floaty McFloatface" (carrying the message "Keep clam and carry on!") will start in the cell outlined in red. Since these cells are in the ocean, the bottles will start to drift. If they were over land, they would not.

Once you've reached this point, you can finish Problem 1 to check your understanding of the input that you are receiving.

Bottle Drift Path

Now that you have a few messages floating about, you'll want to figure out where each is going to end up! The message will continue to drift, based on the currents, until it reaches land. As such, you will need a way to model their float paths. Consider what you'll need to keep track of and identify the best way to store this information.

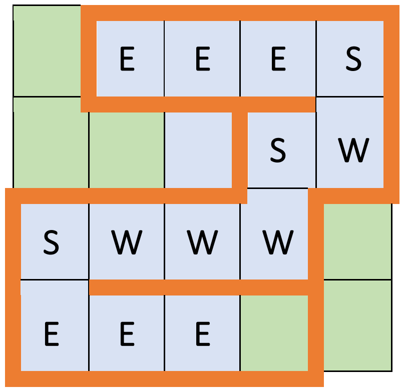
IMPORTANT NOTE: Eventually, we will have to worry about potentially loopy drift paths that loop back onto themselves. You will want to keep this in mind as you plan for how you are storing their drift path.

Behaviour Modeling

Now that you have a model for all of these objects, it is time to model their behaviours! Namely, we're going to start letting our bottle drift the high seas! Assuming the bottle starts its journey in the ocean (i.e. not on land), then it will drift with the current. It will continue to drift until it hits land. At that point, it will stop and (hopefully!) its noteworthy message will be received!

Go with the Flow

Once the bottle is set afloat (or a-flop if on land!), it will continue to follow the flow of ocean surface currents. For instance, our example above will follow the path outlined in orange:



The bottle will start in cell (0,1) and move along with the current until it finally reaches land at cell (3,3). There it will stop and the message will be read! (If it had started on land, the message will still be received, but the bottle will have gone nowhere.)

Tracking the Track

As output, your main method (in Problem 2 & 3) will output the details of the track that your bottle took as it floated along. <<See output examples below>>

WATER YOU DOING?

Problem 0: Sea-ing Clearly

You must begin at the beginning. For this project, you'll want to set the laptop aside and think about how you're going to approach this. Start with your UML diagram. We'll give you an opportunity to upload your diagram in Brightspace. Make a plan, draw it out and upload (jpg, png or PDF) to our A4-UML folder in Brightspace. Remember: An hour of planning often saves 9 hours of doing…

Problem 1: Shell we start?

Time to objectify the Earth (at least a portion of the Earth's surface)! Set up your objects as you've planned things out. We will be looking at how you've structured your work. You'll take in all the input and output a visual map of the Earth's surface, where land is represented by an "X" and ocean cells are represented by the direction of their current.
<<See output examples below!>>

Problem 2: Beach you to it!

At last, you will be able to set your bottle afloat. We're not stopping until we hit land! In this last step, you'll write a program that makes use of all the objects you've created. You'll finally make use of the input files and look to see if you're getting the expected output. All of this will be tested in Codio. <<See output examples below!>>

Problem 3: All tide up!

Last but not least, we will make sure that we don't have any bottles that get stuck in the sea of plastic in the middle of the ocean.[*] If a bottle drift path somehow ends up looping back onto itself (i.e. a location is repeated), then we will note that the bottle is now stuck in a mid-ocean gyre:

```
    Spinner: Starting at (0, 1)
    0: Spinner at (0, 1): In ocean, current taking it E.
    1: Spinner at (0, 2): In ocean, current taking it E.
    2: Spinner at (0, 3): In ocean, current taking it W.
    3: Spinner at (0, 2): <<NOW STUCK IN MID-OCEAN GYRE!>>
```

[*] Note: This is real. In the middle of the Pacific, there is a pile of floating garbage (called the Great Pacific Garbage Patch, or GPGP) that is 1.6 million km^2! Yuck!

Example 1:

Input:

```
3 5
0,0 0,1 1,4 2,3 2,4
E S W S W W W E E E
2
1 1 Bottle A
Greetings from far away lands! Smile often and “seas” the day!
2 0 Bottle B
Keep clam and carry on!
```

Output

Problem 1:

```
XXESW
SWWWX
EEEXX
Bottle named "Bottle A" starting at (1, 1)
Bottle named "Bottle B" starting at (2, 0)
```

Problem 2:

```
Bottle A: Starting at (1, 1)
Bottle B: Starting at (2, 0)
0: Bottle A at (1, 1): In ocean, current taking it W.
0: Bottle B at (2, 0): In ocean, current taking it E.
1: Bottle A at (1, 0): In ocean, current taking it S.
1: Bottle B at (2, 1): In ocean, current taking it E.
2: Bottle A at (2, 0): In ocean, current taking it E.
2: Bottle B at (2, 2): In ocean, current taking it E.
3: Bottle A at (2, 1): In ocean, current taking it E.
3: Bottle B at (2, 3): LANDED!
<<MESSAGE RECEIVED: Keep clam and carry on!>>
4: Bottle A at (2, 2): In ocean, current taking it E.
5: Bottle A at (2, 3): LANDED!
<<MESSAGE RECEIVED: Greetings from far away lands! Smile often and “seas” the day!>>
```

Example 2:

Input:

```
3 3
0,0 1,0 1,1
E E E S W W
1
0 0 Sandy Bottle
Hope you landed safely!
```

Output

Problem 1:

```
XEE
XXE
SWW
Bottle named "Sandy Bottle" starting at (0, 0)
```

Problem 2:

```
Sandy Bottle: Starting at (0, 0)
0: Sandy Bottle at (0, 0): LANDED!
<<MESSAGE RECEIVED: Hope you landed safely!>>
```

Example 3:

Input:

```
3 3
0,0 1,0
S W W N E E N
1
2 0 Soda 'Pop' Seltzer
If there's a will, there's a wave.
```

Output

Problem 1:

```
XSW
XWN
EEN
Bottle named "Soda 'Pop' Seltzer" starting at (2, 0)
```

Problem 2 & 3:

```
Soda 'Pop' Seltzer: Starting at (2, 0)
0: Soda 'Pop' Seltzer at (2, 0): In ocean, current taking it E.
1: Soda 'Pop' Seltzer at (2, 1): In ocean, current taking it E.
2: Soda 'Pop' Seltzer at (2, 2): In ocean, current taking it N.
3: Soda 'Pop' Seltzer at (1, 2): In ocean, current taking it N.
4: Soda 'Pop' Seltzer at (0, 2): In ocean, current taking it W.
5: Soda 'Pop' Seltzer at (0, 1): In ocean, current taking it S.
6: Soda 'Pop' Seltzer at (1, 1): In ocean, current taking it W.
7: Soda 'Pop' Seltzer at (1, 0): LANDED!
<<MESSAGE RECEIVED: If there's a will, there's a wave.>>
```

Example 4:

Input:

```
4 5
0,0 1,0 1,1 2,4 3,3 3,4
E E E S E S W S W W W E E E
2
0 1 Drifter E
Greetings from far away lands! Smile often and “seas” the day!
2 3 Floaty McFloatface
Keep clam and carry on!
```

Output

Problem 1:

```
XEEES
XXESW
SWWWX
EEEXX
Bottle named "Drifter E" starting at (0, 1)
Bottle named "Floaty McFloatface" starting at (2, 3)
```

Problem 2:

```
Drifter E: Starting at (0, 1)
Floaty McFloatface: Starting at (2, 3)
0: Drifter E at (0, 1): In ocean, current taking it E.
0: Floaty McFloatface at (2, 3): In ocean, current taking it W.
1: Drifter E at (0, 2): In ocean, current taking it E.
1: Floaty McFloatface at (2, 2): In ocean, current taking it W.
2: Drifter E at (0, 3): In ocean, current taking it E.
2: Floaty McFloatface at (2, 1): In ocean, current taking it W.
3: Drifter E at (0, 4): In ocean, current taking it S.
3: Floaty McFloatface at (2, 0): In ocean, current taking it S.
4: Drifter E at (1, 4): In ocean, current taking it W.
4: Floaty McFloatface at (3, 0): In ocean, current taking it E.
5: Drifter E at (1, 3): In ocean, current taking it S.
5: Floaty McFloatface at (3, 1): In ocean, current taking it E.
```

```
6: Drifter E at (2, 3): In ocean, current taking it W.
6: Floaty McFloatface at (3, 2): In ocean, current taking it E.
7: Drifter E at (2, 2): In ocean, current taking it W.
7: Floaty McFloatface at (3, 3): LANDED!
<<MESSAGE RECEIVED: Keep clam and carry on!>>
8: Drifter E at (2, 1): In ocean, current taking it W.
9: Drifter E at (2, 0): In ocean, current taking it S.
10: Drifter E at (3, 0): In ocean, current taking it E.
11: Drifter E at (3, 1): In ocean, current taking it E.
12: Drifter E at (3, 2): In ocean, current taking it E.
13: Drifter E at (3, 3): LANDED!
<<MESSAGE RECEIVED: Greetings from far away lands! Smile often and “seas” the day!>>
```

Example 5 (Loopy Example!):

Input:

Only Problem 3:

```
4 5
0,0 1,0 1,1 2,4 3,3 3,4
E E W S E S W S W W W E E E
1
0 1 Spinner
Keep on keeping on!
```

Output

Problem 3:

```
Spinner: Starting at (0, 1)
0: Spinner at (0, 1): In ocean, current taking it E.
1: Spinner at (0, 2): In ocean, current taking it E.
2: Spinner at (0, 3): In ocean, current taking it W.
3: Spinner at (0, 2): <<NOW STUCK IN MID-OCEAN GYRE!>>
```

Example 6 (Another Loopy Example!):

Input:

Only Problem 3:

```
3 3
0,0 1,0
E W S E N N W
2
0 1 Eternal Drifter
Going around again!
2 2 Looper
See you again soon!
```

Output

Problem 3:

```
Eternal Drifter: Starting at (0, 1)
Looper: Starting at (2, 2)
0: Eternal Drifter at (0, 1): In ocean, current taking it E.
0: Looper at (2, 2): In ocean, current taking it W.
1: Eternal Drifter at (0, 2): In ocean, current taking it W.
1: Looper at (2, 1): In ocean, current taking it N.
2: Eternal Drifter at (0, 1): <<NOW STUCK IN MID-OCEAN GYRE!>>
2: Looper at (1, 1): In ocean, current taking it S.
3: Looper at (2, 1): <<NOW STUCK IN MID-OCEAN GYRE!>>
```