The objective of this lab is to solve problems using the Stack ADT

Q1. Write a program that uses the Stack implementation given to you in ***Stack.py*** to **simulate navigation in a web browser.** Every web browser has both a **back** and a **forward button** which allows the user to navigate to **previously** seen web pages.

Your task is to implement this functionality using **two Stack objects called forward_stack and back_stack. The forward_stack and back_stack should be used to store and retrieve the urls as the user navigates the web.**

The interaction with your program will be a sequence of **commands**, either ">", "<", "=", or "x". The ">" and "<" commands will be used to indicate **forward** and **back**, respectively. The "=" command is used to enter a **new url web page**. If the user enters the "=", the computer would expect a **url address**. Finally, if the user enters "x", the program closes. You can **assume** the **web addresses** are always valid.

- You will not actually have to do any web navigation. You will simply be keeping track of the current page by outputting the address of the current page between **"["** and **"]"**.
- There are a **few situations you should take care to handle**.
  - First, if a user enters "<" or ">" without there being a **previous** or **next** page to go to, you should output a **corresponding error** message (as shown in the **sample output** below) but **allow the user to continue browsing**.
  - Secondly, if a user enters **"<"** and then **"="** and then **enters a new web address**, the user's **previous** browsing history in the **">"** direction should be **erased**.
  - For example, if a user is on "**www.google.com**", enters "<", and then enters **"="** and then **"www.yahoo.com**", the history of visiting "**www.google.com**" should be lost. This means that if the user enters **">"**, this should generate an **error message** and not "www.google.com". This behavior can also be seen below in the sample data. You can also test an actual web browser (specifically Firefox or Chrome) if you feel there is other ambiguity. Also, keep in mind that navigation should start with a **home page** which we will set as "**www.cs.ualberta.ca**".
  - The interaction should start with the **current page** being www.cs.ualberta.ca.

Sample Interaction with the program is given below: (Input is in red)

```
Welcome to web browser navigation
Current url: [www.cs.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit: =
Enter a new url: www.google.com
Current url: [www.google.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit: <
Current url: [www.cs.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit <
< is an invalid action
Current url: [www.cs.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit: >
Current url: [www.google.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit: =
Enter a new url: www.yahoo.com
Current url: [www.yahoo.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit: =
Enter a new url: www.ualberta.ca
Current url: [www.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit: =
Enter a new url: www.beartracks.ualberta.ca
Current url: [www.beartracks.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit >
> is an invalid action
Current url: [www.beartracks.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit: <
Current url: [www.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit: <
Current url: [www.yahoo.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit: >
Current url: [www.ualberta.ca]
Enter command: > for forward, < for backward, = for a new url, or x for exit <
Current url: [www.yahoo.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit =
Enter a new url: www.microsoft.com
Current url: [www.microsoft.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit >
> is an invalid action
Current url: [www.microsoft.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit: <
Current url: [www.yahoo.com]
Enter command: > for forward, < for backward, = for a new url, or x for exit url: x
```

Q2. Write a program that uses the Stack implementation given to you in ***Stack.py*** to check if there is **a solution to a maze or not**. If there is a solution, your program shows the message **Goal is reached**, if there is no solution your program shows the message **Goal is Unreachable**.
A simple **maze implementation** is given to you in ***maze.py***.
Hint: You don't need to change anything in file **maze.py**.

To solve a maze using the **Stack**, use the following **algorithm**:

1. Add the **start square** to the stack.
2. Repeat the following **as long as** the stack is **not empty**:
   - **Pop** a square off the stack (the current square)
   - If the current square is the finish square, the solution has been found
   - Otherwise, get the list of squares which can be **moved to** from the current square, and **add** them to the **stack**

Description of **maze.py**:

- The **maze.py** file has two classes: **Maze** and **MazeSquare**. The **Maze** class represents the entire maze and the **MazeSquare** class represents a single square in the Maze
- You can create **a new instance** of **Maze** class by calling the constructor: **my_maze** = **Maze(file_name**), where file_name is the name of a **text file** containing a maze.
- You can get the **start square** of the maze by calling **my_maze.get_start_square()**. This method returns an **object** of type **MazeSquare**.
- To check if a square is the finish square, call **my_maze.is_finish_square(square)**, where square is of type **MazeSquare**.
- The **get_legal_moves()** is a method in the **MazeSquare** class. This method returns a list of objects of type **MazeSquare** that can be **moved to**. Let's say if **square** is an identifier that is bound to an object of type **MazeSquare** then **get_legal_moves()** can be called in the following manner:
  list_of_legal_moves = **square.get_legal_moves()**

An algorithm example is given in the given file MazeAlgorithmExample.gif.

Two **maze files** are included for this exercise: solvable_maze.txt and unsolvable_maze.txt. Check your program using these files. Thus, when you use **solvable_maze.txt** file, your program should find a solution and **print** the message **Goal is reached.** However, when you use **unsolvable_maze.txt** file, your program can’t find a solution and **prints** the message **Goal is Unreachable**. A graphic representation of each maze file is given in the file **MazeExamples.gif**.

.