Computational Thinking and Problem Solving (COMP1002) and Problem Solving Methodology in Information Technology (COMP1001)

Assignment SIX
(Due at noon on **4 December 2020**)

1. [50 marks] Modify A5 Q2. You are allowed to use the sample program in A5 (to be released after A5 submission deadline) or use your own version. Now Alpha has a facing direction represented by, N – '^', E – '>', S – 'v' and W – '<'`. The facing direction is random at the beginning of the program. Based on user input, Alpha can turn left by 90 degrees (L) or turn right by 90 degrees (R). There is also another user option, M, by which Alpha can move one step forward, based on its current facing direction. You have to perform the followings:

   (a) (35 marks) Implement the above requirement. A menu should be provided to allow the user to enter an option and the map will be updated accordingly.
   (b) (15 marks) Add one more user option in the program that allows the user to save the map to a text file. Design how the format of the data is saved. Type your justification as comment at the beginning of the program.

   Your program should look like below:

```
  0 1 2 3 4 5 6 7 8 9
0   v
1
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: M
  0 1 2 3 4 5 6 7 8 9
0
1   v
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: L
  0 1 2 3 4 5 6 7 8 9
0
1   >
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: M
```

```
  0 1 2 3 4 5 6 7 8 9
0
1     >
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: R
  0 1 2 3 4 5 6 7 8 9
0
1     v
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: S
Game saved in gamesave.txt!
  0 1 2 3 4 5 6 7 8 9
0
1     v
2 *   *
3
4         *
5                 *
6     *   *
7               * *
8
9
Move forward (M)
Turn left by 90 degrees (L)
Turn right by 90 degrees (R)
Save the map (S)
Please enter your command: M
```

```
  0 1 2 3 4 5 6 7 8 9
0
1
2 *
3
4         *
5                 *
6     *   *
7               * *
8
9
Alpha crashes!
>>>
```

2. [50 marks] A solution to the MCGW problem can be expressed in different forms, but it is not as simple for human to verify whether a given solution is correct. Here we are to develop a Python program to help verifying the correctness of a given solution. To solve the problem systematically, we are to develop a few functions and break up the original bigger problem into several smaller problems. The following order of breaking down the problem is not the only way, but is a reasonable way. You can assume that the input string is correct in that it does not contain any special character, nor there is a missing of key symbols like "|" and ">".

   (a) (10 marks) The first step to solve the problem is to clean up the input. Develop a Python function **`cleanup`**(solutionStr), which takes as input a string expressing the "solution". This function will remove space, convert all letters into upper case, and split the solution into a list. Provide appropriate comments in your function. Here are some sample outputs from your function:

```
>>> print(cleanup("| MCGW > G | MCW > C | MGW > CW | MG > MCWG |"))
['|MCGW', 'G|MCW', 'C|MGW', 'CW|MG', 'MCWG|']
>>> print(cleanup("|MCGW >MG|CW  >G|MCW >MGW|C >W|MGc >wcm|g >cw|mg >cwmg|"))
['|MCGW', 'MG|CW', 'G|MCW', 'MGW|C', 'W|MGC', 'WCM|G', 'CW|MG', 'CWMG|']
```

   (b) (10 marks) Given a cleaned list from (a) showing the sequence of states of a potential solution, develop a Python function **`check`**(stateList). It would take in a list of states and print out the solution if it is correct. If it is incorrect, indicate the step(s) where there is a problem. There are three types of possible problems: (1) the start and end states are incorrect, i.e., not **|MCGW** and **MCGW|** respectively; (2) a state is illegal, i.e. either the wolf eating the goat or the goat eating the cabbage; (3) a move is not allowed, e.g. man does not move (perhaps the wolf swims or the cabbage "swims"), or the man carries more than one item. Provide appropriate comments in your function. To develop this function, you would need to develop some auxiliary functions as in (c) and (d) first. You may assume that those two functions in (c) and (d) are available to you when you develop this function for (b) at this moment. Here are some sample outputs from your function:

```
>>> print(check(cleanup("| MCGW > G | MCW > C | MGW > CW | MG > MCWG |")))
The move |MCGW -> G|MCW is not allowed
The move G|MCW -> C|MGW is not allowed
The move C|MGW -> CW|MG is not allowed
False
>>> print(check(cleanup("|MCGW >MG|CW  >G|MCW >MGW|C >W|MGc >wcm|g >cw|mg >cwmg|")))
True
>>> print(check(cleanup("MW | CG  > W | MCG")))
Start state MW|CG is incorrect
The state MW|CG is illegal
End state W|MCG is incorrect
False
```

   (c) (10 marks) Develop a Python function **`islegal`**(state) to help in (b). It takes as input a state and returns **`True`** if the state is legal (i.e. goat not eating cabbage and not eaten by wolf) and **`False`** otherwise. Provide appropriate comments in your function. Here are some sample outputs from your function:

```
>>> print(islegal("G|MCW"))
True
>>> print(islegal("MW|CG"))
False
```

(d) (10 marks) Develop a Python function **`canmove`**(state1,state2) to help in (b). It takes as input two states and returns **`True`** if the move is allowed, and returns **`False`** otherwise. Provide appropriate comments in your function. Here are some sample outputs from your function:

```
>>> print(canmove("|MCGW","G|MCW"))
False
>>> print(canmove("G|MCW","C|MGW"))
False
>>> print(canmove("C|MGW","CW|MG"))
False
>>> print(canmove("CW|MG","MCWG|"))
True
```

(e) (10 marks) Complete your Python program by defining **`main`**(solutionList), which takes as input a list of strings, each representing a "solution" and checks for the validity of each "solution". It will *print the solution nicely* if the "solution" is correct as determined by the function **`check`**(stateList) in (b). You may define other auxiliary functions as needed. Provide appropriate comments in your functions and program. Here are some sample outputs from your program:

```
>>> cases = [
"| MCGW > G | MCW > C | MGW > CW | MG > MCWG |",
"|MCGW>GM|CW>G|MCW>GCM|W>C|MGW>CWM|G>CW|MG>MCWG|",
"|MCGW >CM|GW> C|MGW >CWM|G >CW|MG >CWMG|",
"MW | CG  > W | MCG",
"|MCGW >MG|CW  >G|MCW >MGW|C >W|MGc >wcm|g >cw|mg >cwmg|"
]
>>> main(cases)
Input solution: | MCGW > G | MCW > C | MGW > CW | MG > MCWG |
The move |MCGW -> G|MCW is not allowed
The move G|MCW -> C|MGW is not allowed
The move C|MGW -> CW|MG is not allowed
Input solution is incorrect
Input solution: |MCGW>GM|CW>G|MCW>GCM|W>C|MGW>CWM|G>CW|MG>MCWG|
Input solution is correct
The solution:
man and goat to West
man to East
man and cabbage to West
man and goat to East
man and wolf to West
man to East
man and goat to West
Input solution: |MCGW >CM|GW> C|MGW >CWM|G >CW|MG >CWMG|
The state CM|GW is illegal
Input solution is incorrect
Input solution: MW | CG  > W | MCG
Start state MW|CG is incorrect
The state MW|CG is illegal
End state W|MCG is incorrect
Input solution is incorrect
Input solution: |MCGW >MG|CW  >G|MCW >MGW|C >W|MGc >wcm|g >cw|mg >cwmg|
Input solution is correct
The solution:
man and goat to West
man to East
man and wolf to West
man and goat to East
man and cabbage to West
man to East
man and goat to West
```

Here are some test cases that you generated together:

- `| MCGW > G | MCW > C | MGW > CW | MG > MCWG |`
- `|MCGW>GM|CW>G|MCW>GCM|W>C|MGW>CWM|G>CW|MG>MCWG|`
- `|MCGW >C|MGW >CW|MG >CWMG|`
- `| mcgw> c| mgw> cw| gm> cwmg |`
- `|MCGW >CM|GW> C|MGW >CWM|G >CW|MG >CWMG|`
- `MW | CG > W | MCG`
- `|MCGW >MG|CW >G|MCW >MGC|W >MGCW|`
- `MGC|W > C|MGW > MWC|G > WC|MG > MGCW|`
- `| MCGW>MG | CW>G| MCW>MCG | W>C | WMG>MWC | G>MWCG |`
- `MCGW |>CW | MG>MCW | G>W | MCG>MWG | C>G | MWC>MG | WC>| MGWC`
- `|MCGW >MG|CW >G|MCW >MGW|C >W|MGc >wcm|g >cw|mg >cwmg|`

Submission Instructions

Follow the steps below:

1. Create a folder and name it as <student no>_<your name>,
   e.g., **`12345678d_CHANTaiMan`**
2. For Q1 and Q2. You need to submit the source file (**`.py`**). Name the **`.py`** files as A6_Q<question no>_<student no>_<your name>**`.py`**,
   e.g., **`A6_Q1_12345678d_CHANTaiMan.py`**
3. Put all the **`.py`** files into the folder.
4. Compress the folder (**`.zip`**, **`.7z`**, or **`.rar`**).
5. Submit the file to Blackboard.

A maximum of **3 attempts** for submission are allowed. **Only the last attempt will be graded**. A late penalty of 5% per hour will be imposed.