# CSE 241 Programming Assignment 2

## DUE

March 23, 2018, 23:55

## Description

- This is an individual assignment. Please do not collaborate
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

This assignment is about designing, implementing and testing classes for the following setup:

- A small version of the game called Tsuro will adapted for the command line interface.

#### Tsuro

- Read the explanations about the original game (https://en.wikipedia.org/wiki/Tsuro)

#### Basic Elements of the Game

#### Tokens

- Tokens represent players.
- For the version you are going to implement, tokens will be displayed as single characters.

#### Path Cards

- Square cards which have 2 ports on each of their sides and various routing information.
- You can display a Path card as follows:

```
....1..2....
:   8  6   :
8   1  5   3
7   4  7   4
:   2  3   :
....6..5....
```

- Here, ports are numbered (1,2,3,4,5,6,7,8).
- Inside, the routing information is presented. For the particular card above:
  - port 8 is connected to port 1
  - port 7 is connected to port 4
  - port 6 is connected to port 2
  - port 5 is connected to port 3
- Connections are not directed.
- A path card can be rotated clockwise or counter-clockwise(repeatedly)

#### 2D grid

- This game will be played on a 2D grid.
- Each cell on this grid will either be occupied by a path card or it will be empty.
- Cards can only be placed in empty cells
- Below is an example of a partially filled 3x3 grid

```
------o--o--------o--o--------o--o------
| ....1..2........7..8....              |
| :   8  6   ::   1  6   :             |
o 8   1  5   36   8  7   1             o
o 7   4  7   45   3  4   2             o
| :   2  3   ::   2  5   :             |
| ....6..5........4..3....             |
| ....3..4....                         |
| :   4  3   :                         |
o 2   1  6   5                         o
o 1   2  5   6                         o
| :   7  8   :                         |
| ....8..7....                         |
| ....1..2........1..2........1..2.... |
| :   5  6   ::   6  5   ::   8  6   : |
o 8   4  7   38   3  8   38   1  5   3 o
o 7   3  8   47   4  7   47   4  7   4 o
| :   2  1   ::   1  2   ::   2  3   : |
| ....6..5........6..5........6..5.... |
------o--o--------o--o--------o--o------
```

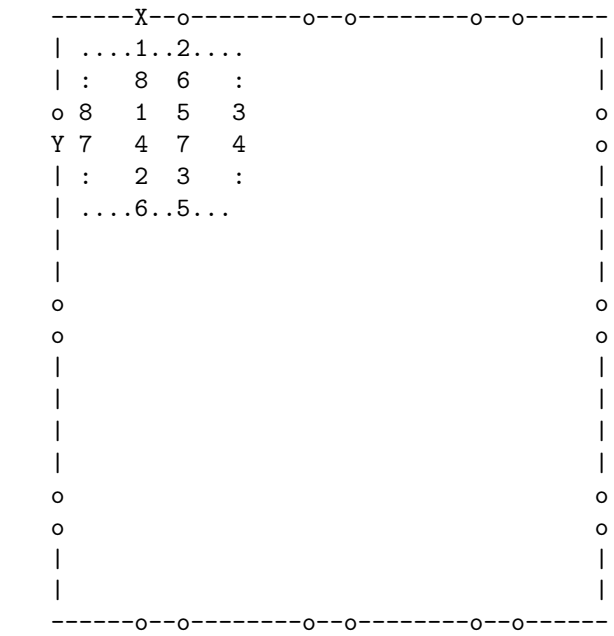
- On the sides of the grid `'o'`s represent exit/entry ports of the game.
- A User token may start at any of the exit/entry ports and exit from any of the ports.
- If the ports of path cards(tiles) collide, they are assumed to be connected. For example, port 3 of the tile 1 (first row, first column) is connected to port 6 of the tile on the right.

**Rules of The Game**

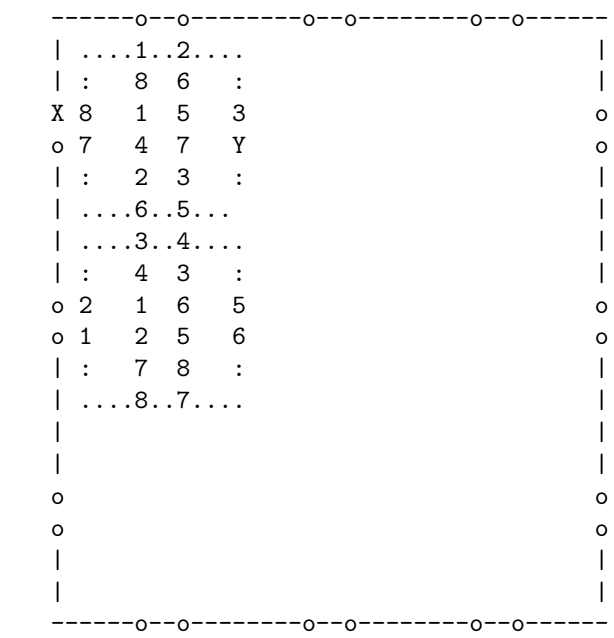
- For a general understanding of the game please read the necessary wikipedia articles. You can find more information on other webpages.
- Basically:
  - It is a turn-based game.
  - There are at least 2 players, each one has a token.
  - Game starts randomly. (random entry ports are chosen for each player)
  - Initially, the 2D grid is empty.
  - Each player is given a set of cards(Generally 3 cards.)
  - A turn starts with the player deciding on a card(selecting a card from the set given). If the set of cards is empty, a new set is requested (how you generate a set is up to you. It has to be flexible for further modifications. So hide unnecessary details.)
  - Once a card is selected, it is placed in an empty cell on the 2D grid.
  - The token follows the route on the card placed.
  - The objective is to remain on the game.

## Example:

- Initial. Player1(token X) places a path card:




- With the tile which is recently placed, the game advances. X follows the path and exist the game(loses). Y follows its path and waits for the next event.
- It's Player2(token Y)'s turn:




- At this stage there isn't any other player, so Y wins.

## Implementation

- You have to use OOP principles.
- You cannot use the topics we haven't covered in class. (you can utilize the tools you know from C programming course)
- For the solution, you may need to create several classes.

- Try to use the concepts such as encapsulation, protecting variables with `const` keyword, overloading functions/operators,
- Implement the whole game as a class and write necessary constructors for the cases such as player vs player, 3 players against each other etc... (for this you may need to have a `player` class as well.)
- This document does not fully describes the visualization of the game flow. Create necessary dialogs such as displaying set of cards to the user, asking user choice, presenting the state of the game, etc... It is your responsibility to create an appropriate flow so that the game can be played.

## Remarks

- Write comments in your code.
- If your code does not compile you will get `0`
- Do not share your code with your classmates.

## Turn in:

A complete C++ program `<assignment_2_fullname_id.cpp>` which can be compiled using the following command:

```
g++ -std=c++11 assignment_2_fullname_id.cpp -o assignment_2_fullname_id
```

If your program requires additional compile and link options, state those requirements at beginning of your source code as comment.

## Late Submission

- `(0,24]` hours: `-20%`
- `(24,48]` hours: `-40%`
- `(48,72]` hours: `-60%`
- `(72,-)` hours: `-100%`