

# Tic-Tac-Toe Stage

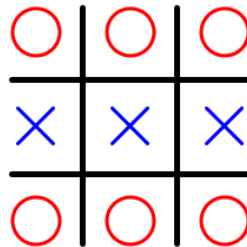
**\*This homework is a programming project.**

**\*You need to submit your Racket file and png file on Blackboard.**

**\*While working on the project, read this document THOROUGHLY.**

## Project Description

In this project, you will write a Racket program to draw a tic-tac-toe stage like the below.

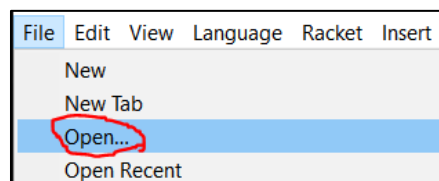


## Project Guideline

Follow the below guideline to write a tic-tac-toe program.

### Part 0. Setup Programming Environment

1. Download the required file (`tictactoe.rkt`) from Blackboard and save it under any folder you want.
2. In DrRacket, open `tictactoe.rkt`.



3. Inside `tictactoe.rkt`, you will see a basic structure of a tic-tac-toe program with some pre-defined functions. Here are short descriptions of the pre-defined functions.

`whitebox`: it draws a white box.

`transpbox`: it draws a transparent box.

`pensetup`: it set up a pen with desired color and width. This function is used to set up the color and width of the line. You can input the color and width like (`pensetup "red" 3`) and (`pensetup "black" 5`)

## Part 1. 0 mark

In this part, you will define a function **omark** that draws O in the result window.

1. Define a function **singlecircle** that draws a circle using **circle** function. It needs to draw a circle that satisfies the below features.

radius: 20  
color: red  
line's width: 3

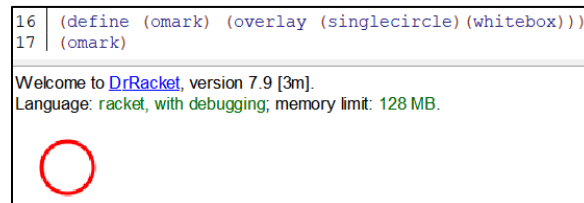
To draw a circle satisfying the above features, you can use a **circle** function like

```
(circle 20 "outline" (pensetup "red" 3))
```

2. Define **omark** function as shown below where it overlays **singlecircle** on top of **whitebox** using **overlay** function.

```
(define (omark) (overlay (singlecircle)(whitebox)))
```

3. When you call **(omark)** (which means that you write **(omark)** in the program and run it), it should draw a single red circle in the result window as shown below.



## Part 2. X mark

In this part, you will define a function **xmark** that draws X in the result window.

1. Define a function **singleline** that draws a single line of X mark using a **line** function. It needs to draw a line that satisfies the below features.

x of end-point: 30  
y of end-point: 30  
color: blue  
line's width: 3

You can set up a color and a width of a line using **pensetup** function like we did for a circle in step 1 of Part 1. **pensetup** function expression will be the last argument of **line** function. (**line** function follows a syntax (**line** <x> <y> <pen or color>).)

2. Define a function **flipsingleline** that flips **singleline**. To flip a line image, you can use either **flip-horizontal** or **flip-vertical** function.
3. Define a function **xlines** that overlays **singleline** on top of **flipsingleline**. To overlay one image on top of another image, you can use **overlay** function as shown in step 2 of Part 1.


4. Define `xmark` function as shown below where it overlays `xlines` on top of `whitebox`.

```
(define (xmark) (overlay (xlines)(whitebox)))
```

5. When you call `(xmark)`, it should draw a single blue X in the result window as shown below.

```
23 (define (xmark) (overlay (xlines)(whitebox)))
24 (xmark)
```

Welcome to [DrRacket](#), version 7.9 [3m].  
Language: [racket](#), with [debugging](#); memory limit: 128 MB.



### Part 3. Whole board with Os and Xs

In this part, you will define a function `wholeboard` which arranges Os and Xs vertically and horizontally as shown in the final result on the first page of this document.

If you want to arrange marks horizontally, you can use `beside` function. If you want to arrange marks vertically, you can use `above` function.

For example:

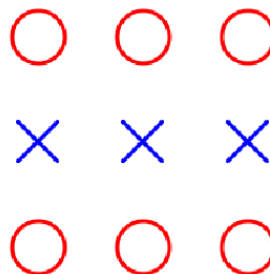
`(beside (omark) (omark) (omark))` will give the below result.



`(above (omark) (omark) (omark))` will give the below result.



You need to figure out how to define `wholeboard` function using `beside` and `above` functions so that you have the below result when you call `(wholeboard)`.



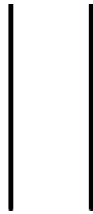
## Part 4. Grid

In this part, you will define a function `grid` to draw a grid image.

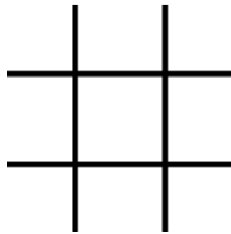
1. Define a function `vline` that draws a single line using `line` function satisfying the below requirements.

x of end-point: 0  
y of end-point: 200  
color: black  
line's width: 5

2. Define a function `vertline` that overlays `vline` on top of `transpbox`. Since `vertline` will be overlayed on top of `wholeboard`, if you use `whitebox` instead of `transpbox`, some part of `wholeboard` will not be visible later.
3. Define a function `vertgrid` that arranges two `vertlines` horizontally. You can use `beside` function. By calling `(vertgrid)`, you will have the below result.



4. Define a function `horigrid` that arranges two horizontal lines vertically. Instead of drawing two horizontal lines, you can simply rotate `vertgrid` by 90 degrees. (You can use `(rotate 90 <image>)` to rotate given `<image>` by 90 degrees.)
5. Define a function `grid` that overlays `vertgrid` on top of `horigrid`. By calling `(grid)`, you will have the below result.



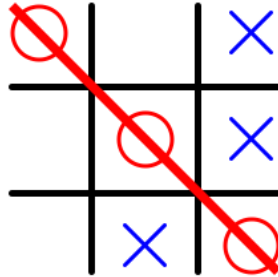
## Part 5. Tic-tac-toe stage

1. Define a function `tictactoe` that overlays `grid` on top of `wholeboard` like the below.

```
(define (tictactoe) (overlay (grid) (wholeboard)))
```

2. When you call `(tictactoe)`, you will have the final result.
3. Call `(save-image (tictactoe) "tictactoe.png")`. Then you will have 'tictactoe.png' file in the folder where you have the tic-tac-toe program. You need to submit the png file with the Racket source code (`tictactoe.rkt`) on Blackboard.

**Bonus Question.** Modify the tic-tac-toe program in a way that you can have the below result which looks like a ‘real’ tic-tac-toe stage. You still need to submit a png file of this bonus question to get a bonus point (2 pts).



**Submission Guideline**

1. Upload `tictactoe.rkt` and `tictactoe.png` on Blackboard.
2. After you submit them, DOUBLE-CHECK whether you’ve submitted the correct files on Blackboard.