ACTION PERFORMED - else CHESS

```
else
{ int n = ....
  int x = ....
  int y = ....
```

global; starts as int last = -1

check #1 invalid

```
if (turn != colour[x][y] && last == -1)
    it's not your turn.
```

A dialog box, showStatus ?

click #1 - valid

```
else if (last == -1 && turn == colour[x][y])
{   if (piece[x][y] == 'P')
        selectPawn(x,y);
    and so on for all pieces
        last = n;
}
```

N → if (piece[x][y] == 'N') selectKnight(x,y);
R
B
Q
K

click #2 - handles valid & invalid

```
else
{ int lastx = last/col;
  int lasty = last%col;
```

click #2, valid move

```
  if (select[x][y] == 'S')
  { move
    switch turn
  }
```

piece[x][y] = piece[lastx][lasty];
piece[lastx][lasty] = '0';
colour[x][y] = colour[lastx][lasty];
colour[lastx][lasty] = '0';

— in turn sheet on website

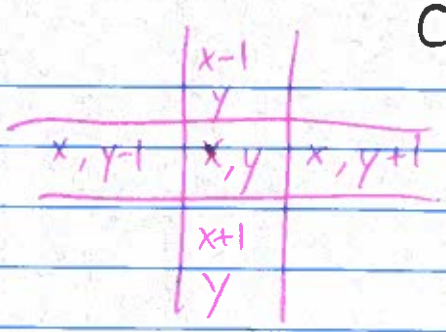
```
    reset select array to 'U'
    last = -1;
}
redraw();
```

```
for(int i=0; i<row; i++)
{ for(int j=0; j<col; j++)
  { select[i][j] = 'U';
  }
}
```

} end else.

CHESS

PAWN



In AP, add the PAWN

```
if (piece[x][y] == 'P')
    selectPawn(x,y);
```

Make the method

```
public void selectPawn (int x, int y)
{
    if(colour[x][y]=='W' && x==6)
    {  select[x-1][y] = 's';
       select[x-2][y] = 's';
    }
    else if (colour[x][y]=='B' && x==1)
    {  select[x+1][y] = 's';
       select[x+2][y] = 's';
    }
    else if (colour[x][y]=='W')
    {
       select[x-1][y]='s';
    }
    else if (colour[x][y]=='B')
    {
       select[x+1][y]='s';
    }
```

W moves up. 6 is the start row. it can move 2 spaces

B moves down. 1 is it's start row

other places of W moves 1 space up

other places of B moves 1 space down.

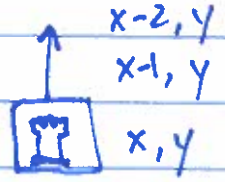
now, there's more ... the kill conditions are missing as are the reaching the other side and becoming a queen ... but this is a start.

```
}
```

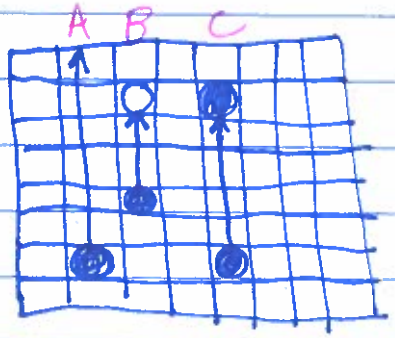
Rook

CHESS

Let's start with UP.



the x values or rows change as we more up.



The 3 conditions we need to handle are:

A) we highlight until we hit the end of the board. No pieces in way.

B) we highlight until we hit the enemy AND the enemy too.

c) we highlight until we hit ourself we don't highlight that piece.

In AP add the Rook

```
if (piece[x][y] == 'R')
    selectRook (x,y);
```

Make the method:

```
public void selectRook (int x, int y)
{

}
```

inside the method, add in the up code:

```
//Up
int cx1 = x-1;          // move up one spot; make a variable to hold position
while (cx1>=0 && colour[cx1][y] == '0')   // we haven't fallen off end; it is still blank
{
    select[cx1][y] = 'S';   // select it.
    cx1--;                  // keep going up.
}
// at this point, we've hit a piece or the edge.
if (cx1>=0 && colour[cx1][y] != colour[x][y]   // not edge; it's not us
    && colour[cx1][y] != '0')                  // it's not blank either
{
    select[cx1][y] = 'S';   // since it isn't us, or the edge, it is our enemy. Select it too. They can kill it!!
}
```

Loop to highlight up

Kill condition

(or whatever your Blank is.)

up ↑

cx1-3, y
cx1-2, y
cx1-1, y
cx1, y

Now: TEST IT.
- Remove the pawn in front
- move it around.

Then: Add Down
Then Right
Then Left.
Test each.