The “chess” game – 4 points

You have to write a program that when given an input integer for the board size, knight location, number of pawns and their location, generates a chess board with the given size and places the pieces according to their location. Example:

Given:
10
k 5 5
4
1 1
8 8
1 8
8 1

Print:

```
_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ k _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _
```

Your task is to implement the movement of the chess knight. Move the knight according to user input at each turn, and allow for the taking of the pawns (they will NOT move). The input should be in the form of two not separated characters:

ul – up left
ur – up right
ru – right up
rd – right down
dl – down left
dr – down right
lu – left up
ld – left down

After each turn the program should update the playing field and print it to reflect the move made by the knight.

The game ends when all pawns are taken, at the end of the game you have to print how many turns were made and what they were.

Your program should be divided **AT LEAST** in 4 parts – “main”, “read input and generate field”, “process turn” and “print field”.

Bonus 1 – 2 points

Update the program to be able to directly take input from a .txt file and generate the field that way. How many moves it took to end and what they were should be written to a result.txt file.

Bonus 2 – 2 points

Update the program to be able to generate a playing field only from the size of the field and number of pawns given. Knight and pawns should be placed in random spots on the board.

Bonus 3 – 4 points

Update the program to be able to replace the knight with one of two pieces – queen and rook, depending on the input. The pieces should play like in the game chess, the input to move them should be similar to the knight's. When implementing this point you have to have a logical part for the program for each of the 3 player pieces. Example:

Given:
10
q 5 5
4
1 1
8 8
1 8
8 1

Print:

_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ q _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _

Given:
10
r 5 5
4
1 1
8 8
1 8
8 1

Print:

_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ r _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ p _ _ _ _ _ _ p _
_ _ _ _ _ _ _ _ _ _

Queen input:
r 5 – right 5 squares
u 5 – up 5 squares
l 5 – left 5 squares
d 5 – down 5 squares
ur 5– up-right diagonal 5 squares
ul 5 – up-left diagonal 5 squares
dr 5 – down-right diagonal 5 squares
dl 5 – down- left diagonal 5 squares

Rook input:
r 5 – right 5 squares
u 5 – up 5 squares

l 5 – left 5 squares
d 5 – down 5 squares

Bonus 4
4 points for Knight
2 points for rook
3 points for queen

Implement the option for the player to choose not to play and for the program to find a path to win the game for them – it should still print how many turns it took and what they were, but it's not needed for it to print the board at each turn.

Bonus 5
5 points for Knight
5 points for rook
5 points for queen

Extend bonus 4 to find how to win in the shortest amount of turns.