Question 1

Write a program `WordCount.java` that takes contents of a file and list the five most frequently used words in it.
Your program should prompt user for a filename and - assuming that a file with specified filename exists - attempt to read the file and process its content.
Following is a sample run of the program to illustrates the expected output.

```
$ javac Word.java WordCount.java
$ java WordCount
Enter Filename: testFile.txt
Most Frequent Words:
word            frequency
the             14
and             9
they            6
fence           4
went            4
```

Where content of the file `testFile.txt` is shown below.

> Through the fence, between the curling flower spaces, I could see them hitting. They were coming toward where the flag was and I went along the fence. Luster was hunting in the grass by the flower tree. They took the flag out, and they were hitting. Then they put the flag back and they went to the table, and he hit and the other hit. Then they went on, and I went along the fence. Luster came away from the flower tree and we went along the fence and they stopped and we stopped and I looked through the fence while Luster was hunting in the grass.

Note that words "*They*" and "*they*" are considered as one word. In addition, the program is not sensitive to punctuations and regards the words "*fence*", "*fence,*" and "*fence.*" as three occurences of the same word.
A template for how to open, read and close files in Java is provided in the file `FileHandleSample.java` that accompanies this assignment.

Question 2

Stella, the owner of a local diner has a box of magnetic characters that she uses to put different messages on the sign board of her establishment.
At times when Stella runs out of characters for her desired messages, she gets annoyed and wishes if there was an easy way to test if a message can make it to the sign board.
Write a program `ShopSign.java` that prompts user to first enter a string that represents the container of magnetic characters and then another string that represents the proposed message.
You can assume magnetic characters are only a set of lowercase and uppercase english alphabets, numbers and exclamation mark.
Following is an expected sample run of your program.

```
$ java ShopSign
Enter available characters:
AAABCCCDDDEEFGHIILLMMNNOOPQRSSSSTTTTUUVWXYY,,!!
Enter proposed message: HOT DOGS AND BUNS!
The message makes it to the sign board.
$ java ShopSign
Enter available characters:
AAABCCCDDDEEFGHIILLMMNNOOPQRSSSSTTTTUUVWXYY,,!!
Enter proposed message: HOT DOGS AND SANDWICHES!
We are short of character H.
```

Question 3

You are asked to develop a simple ticket printing machine for box-office of a small cinema with a total of 50 seats, 10 seats in each row. Ticketing procedure is based on a few simple policies as follows:

1. Rows closer to screen are always prioritized.
2. Seats in a row are filled from the leftmost to the rightmost.
3. All seats assigned to a group of people must be in the same row.

Write a program `BoxOffice.java` that prompts ticket seller for number of tickets to print, showing the number of total available seats at the same time. If sufficient number of seats are available in a single row, your program should print the list of seats assigned to the group. Otherwise, it should show a warning indicating tickets are not available for the group.
Program terminates when all tickets are sold and no more seats are available. The following sample run illustrates the expected output of your program.

```
$ javac BoxOffice.java
$ java BoxOffice
50 seats still available.
Number of tickets to print? 6
Sold seats 1-6 of row 1

44 seats still available.
Number of tickets to print? 5
Sold seats 1-5 of row 2

39 seats still available.
Number of tickets to print? 0

39 seats still available.
Number of tickets to print? 11
Tickets not available for group of 11.
```

Question 4

Write a program `SnakeGame.java` that implements a simplified version of the fameous snake arcade game where user controls movement of a snake towards the food source within a matrix of size 5×10.
The syntax for user commands is `[number]<direction>` where `direction` is one of a, s, d or w letters corresponding to movement in left, down, right

and up directions and `number` is the number of cells to move in the specified direction. As an example, command `3d` will move the snake three cells to the right. In case the destination cell is outside the matrix, the command will be rejected and the snake will not move at all. In case the number is not provided, the snake will move one cell in the specified direction.
The initial position of snake is randomely generated. Once the snake arrives at the cell where the food source is, the food source is considered as consumed and a new food source will appear at a new randomly generated location. The program terminates when the snake eats five food sources.
For simplicity, following assumptions can be made in your design:

1. The snake has size one and it does not grow in size when it feeds.
2. There is no barrier inside the box and both the food and the snake can be in any of the 100 cells.
3. At any moment, there is only one food source present on the board.

Following is a portion of expected output of a sample run of the program.

```
--------------------
Points to Eat: 5

      o
   |


Enter command: 4d
--------------------
Points to Eat: 5

      o
      |


Points to Eat: 4
--------------------
Enter command: 1w
  o
      |


Enter command: 2w
--------------------
```

```
Invalid command.
Points to Eat: 4
  o
       |



Enter command: 5a
```