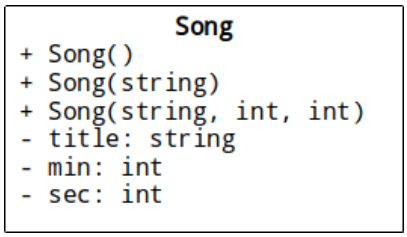
CENG241 Labwork 6

1. Implement the following class:



   The default constructor should set all private variables to empty values. The first overloaded constructor should get song title via parameter but set song duration to 0:00. The second overloaded constructor should get all private variables via parameters. Implement getters and setters for all private variables.

   Write a program in which you demonstrate usage of all constructors by creating three Song objects. Use setters together with the default and first overloaded constructor. Use getters to display the playlist and to calculate and display total duration of the playlist at the end of the program.

   Hint: You may include `iomanip` in your program and use `std::setw` and `std::setfill` for displaying seconds properly (e.g. "3:09" instead of "3:9").

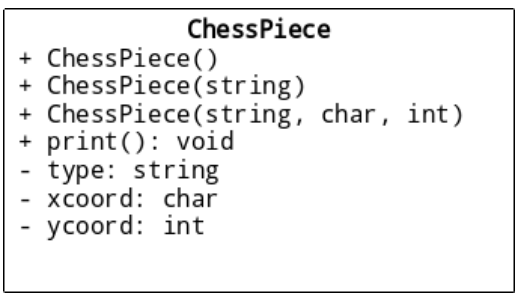
```
Sample Run:
Enter title and duration of first song: The Bard's Song
3 09

Enter title and duration of second song: Children of the Elder God
3 40

Enter title and duration of third song: How You Remind Me
3 49

Your playlist:
- The Bard's Song (3:09)
- Children of the Elder God (3:40)
- How You Remind Me (3:49)
Total duration: 10 minutes, 38 seconds.
```

2. Implement the following class:



   The default constructor should set type of the piece and its coordinates randomly, therefore no user input is necessary. The first overloaded constructor should get piece type as parameter but set its coordinates randomly. The second overloaded constructor should set type of the piece and its coordinates via parameters. Initialize random variables in constructors via assignment and use

initializer lists for non-random variables. print() must print type of the piece and its coordinates on screen. Feel free to implement getters and setters as necessary.

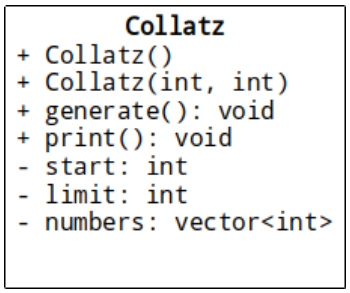
Write a program in which you demonstrate usage of all constructors. Do not forget to free dynamic objects and variables where appropriate (if used any).

```
Sample Run #1:
Enter type for second piece: pawn
Enter type and coordinates for third piece: king b 7
First piece: knight is at g2
Second piece: pawn is at d2
Third piece: king is at b7

Sample Run #2:
Enter type for second piece: bishop
Enter type and coordinates for third piece: rook a 8
First piece: queen is at b5
Second piece: bishop is at c3
Third piece: rook is at a8
```

3. *"The Collatz conjecture is a conjecture in mathematics that concerns sequences defined as follows: start with any positive integer n. Then each term is obtained from the previous term as follows: if the previous term is even, the next term is one half of the previous term. If the previous term is odd, the next term is 3 times the previous term plus 1. The conjecture is that no matter what value of n, the sequence will always reach 1."* - Wikipedia (`https://en.wikipedia.org/wiki/Collatz_conjecture`

   Implement the following class:



   The default constructor should set first term to a random number between 10 and 50 and maximum number of terms to 10, while the overloaded one receives the first term and maximum number of terms as parameters. Use initializers on constructors. generate() must fill the numbers vector according to the rules above and stop when either "1" is generated, or maximum number of terms has been reached. print() must display the sequence on screen. Feel free to implement getters and setters as necessary.

   Write a menu-driven program where the user specifies whether a default or a custom (overloaded) sequence object be created for 5 Collatz sequences and printed on screen. Do not forget to free dynamic objects and variables where appropriate (if used any).

```
Sample Run:
1. Default (start: random, limit: 10)
2. Custom
Your choice: 1
Your choice: 1
Your choice: 2
Enter start number and limit: 47 600
Your choice: 1
Your choice: 2
Enter start number and limit: 341 15

Collatz #1: 48 -> 24 -> 12 -> 6 -> 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> end
Collatz #2: 21 -> 64 -> 32 -> 16 -> 8 -> 4 -> 2 -> 1 -> end
Collatz #3: 47 -> 142 -> 71 -> 214 -> 107 -> 322 -> 161 -> . . . -> 4 -> 2 -> 1 -> end
Collatz #4: 25 -> 76 -> 38 -> 19 -> 58 -> 29 -> 88 -> 44 -> 22 -> 11 -> end
Collatz #5: 341 -> 1024 -> 512 -> 256 -> 128 -> 64 -> 32 -> 16 -> 8 -> 4 -> 2 -> 1 -> end
```