Practical Test Questions

Q1:

Given a farm with cattle, sheep and goats. Calculate the optimal number of each animal provided the following parameters and restrictions.

Parameters and Restrictions: Each animal brings in a certain amount of money, eats a certain amount of food and drinks a certain amount of water per unit respectively. The farm only has a certain amount of food and water that they can provide for all the livestock.

The optimal amount of animals is defined as the amount of goats, sheep and cattle that maximise profit whilst staying within food and water restrictions

Input for program:

Pc Ps Pg //Profit for cows, sheep and goats

F //Food restrictions for the farm

Fc Fs Fg //Food consumed per cow, sheep and goat

W //Water restriction for the farm

Wc Ws Wg //Water consumed per cow, sheep and goat

Output supplied by program:

nc ns ng //Optimal number of cows, sheep and goats

Note: These must be whole amounts (8 as opposed to 8.3).

Sample I/O:

Sample Input:

```
18.01 1.05 0.95

28.20

2.00 8.00 8.05

24.91

1.00 12.04 16.89
```

Sample Output:

```
14 0 0
```

Test Cases:

1)

```
The expected output was:
0 0 30

Input supplied to your program:
7.77 4.3 8.51
76.73
7.04 3.19 1.25
94.39
3.32 8.77 3.05
```

2)

```
The expected output was:
10 0 1

Input supplied to your program:
7.57 1.43 8.15
32.49
2.26 1.19 6.89
86.02
7.48 9.86 8.56
```

3)

```
The expected output was:
0 0 5

Input supplied to your program:
8.03 2.29 8.63
55.39
2.97 3.47 7.26
30.86
7.32 2.48 5.91
```

4)

```
The expected output was:
0 13 0

Input supplied to your program:
4.69 7.92 2.33
89.02
5.96 6.58 7.0
73.36
8.31 2.67 3.58
```

Q2:

Introduction:

Modulo number lines are number lines where the upper and lower limits are defined by the modulus rather than the typical limits. One common modulo number line is that of the wall clock. For hours of the clock past 12, we say that the time is PM; whilst for hours before 12, we say that is AM. In other words 2 hours past 12 is defined as 2PM. In essence the number line wraps around itself.

Another way of writing what was illustrated above is: (12+2) % 12 = 2. This can be done for multiplication too (12*2) % 12 = 12, which would be the equivalent of one day.

The Problem:

Given a modulus limit n (where n acts as the upper bound for the timeline) and a number x, find all possible square roots of x within the given limit.

I.e. $a^2 \equiv x \bmod n$ [for $0 < a < n$].

(HINT: The way I solved this was to rearrange the equation)

Input:

```
n       //Limit

x       //X
```

Output:

```
A1 A2 A3 A4 //Where Ai is a root of x
```

Sample I/O:

Sample Input:

```
16

4
```

Sample Output:

```
2 6 10 14
```

Test Cases:

1)

```
The expected output was:
13716 18201

Input supplied to your program:
31917
9858
```

2)

```
The expected output was:
2444 20450

Input supplied to your program:
22894
20696
```

3)

```
The expected output was:
3272 10395 22706 29829

Input supplied to your program:
33101
14361
```

4)

```
The expected output was:
4801 19439 25601 32951 40239 47589 53751 68389

Input supplied to your program:
73190
67941
```