# CS1010E: Programming Methodology

## PE1 (2019/2020 Sem 2)

### Files

- `PE1.pdf`
- `Question1.py`
- `Question2.py`
- `Question3.py`

### Coursemology

- `Past PE1 > CS1010E 2019/20 Sem 2`

### Questions

## Question 1: She Loves Me, She Loves Me Not

Valentino is in love with his friend from CS1010E. However, he is not sure if she likes him back or not. Leading up to Valentine's day, he devised a method to get her attention by giving her one flower, once a day, until Valentine's day.

So Valentino bought a bouquet of flower and arranged them in circle. There are three types of roses: `"R"` for red roses, `"P"` for pink roses and `"W"` for white roses. The top-most flower is given an index 0 such that the bouquet can also be arranged as a line. Consider the following bouquet represented in Python as a tuple (`tuple`):
`bouquet = ("R", "P", "W", "W", "P", "R", "R", "R")`

To pick the flower, he first pick a number `k`. Every day, he will choose the `k`-th roses from the current position, he then removes the rose and gives it to his friend. So, if `k = 3` with the `bouquet`

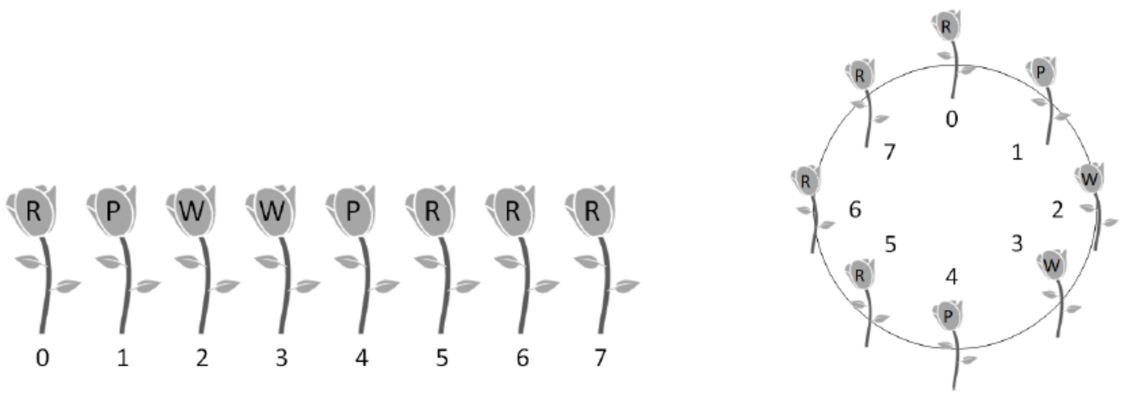


Figure 1: Visualisation of `bouquet = ("R", "P", "W", "W", "P", "R", "R", "R")` as line (*left*) and as circle (*right*)

.

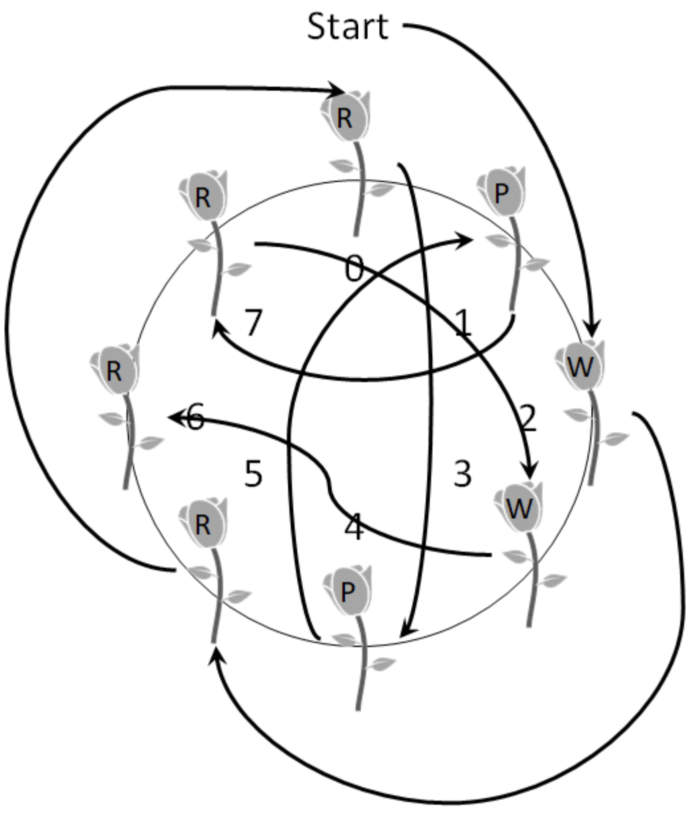
above, the 3rd rose is the rose at index 2 which is the white rose.

This process continues until there are no more roses. However, it continues from the location of the removed roses. Note that once the rose at index 2 is removed, the state of the `bouquet` will be: `("R", "P", "W", "P", "R", "R", "R")`. The 3rd rose from the removed rose is the red rose at index 4 (*i.e.*, the new index 4 after the removal of the white rose, it corresponds to index 5 of the original `bouquet`). Removing this red rose, we get `("R", "P", "W", "P", "R", "R")`.

What we are interested is what roses are given by Valentino. Given `bouquet = ("R", "P", "W", "W", "P", "R", "R", "R")` and `k = 3`, we have the following sequence:

| `bouquet` | **Rose** | **Explanation** |
|---|---|---|
| `("R", "P", "W", "W", "P", "R", "R", "R")` | `"W"` | 3rd rose from beginning |
| `("R", "P", "W", "P", "R", "R", "R")` | `"R"` | 3rd rose from removed rose |
| `("R", "P", "W", "P", "R", "R")` | `"R"` | 3rd rose from removed rose, loop back to 1st rose |
| `("P", "W", "P", "R", "R")` | `"P"` | 3rd rose from removed rose |
| `("P", "W", "R", "R")` | `"P"` | 3rd rose from removed rose |
| `("W", "R", "R")` | `"P"` | 3rd rose from removed rose, loop back to 1st rose |
| `("W", "R")` | `"R"` | 3rd rose from removed rose |
| `("R",)` | `"W"` | 3rd rose from removed rose, loop back to 1st rose |
| `()` | `"R"` | The only rose remaining |

The sequence of roses given can then be summarised as the string `"WRRPPPRWR"` by taking the rose from top to bottom. The visualisation of the selection is given below:



## 1.1 Rotating [5 marks]

To help with the entire process, we can use the circular view of the bouquet and write a function to rotate the bouquet by `step` number of steps. Consider the bouquet:

```
("R", "P", "W", "W", "P", "R", "R", "R")
```

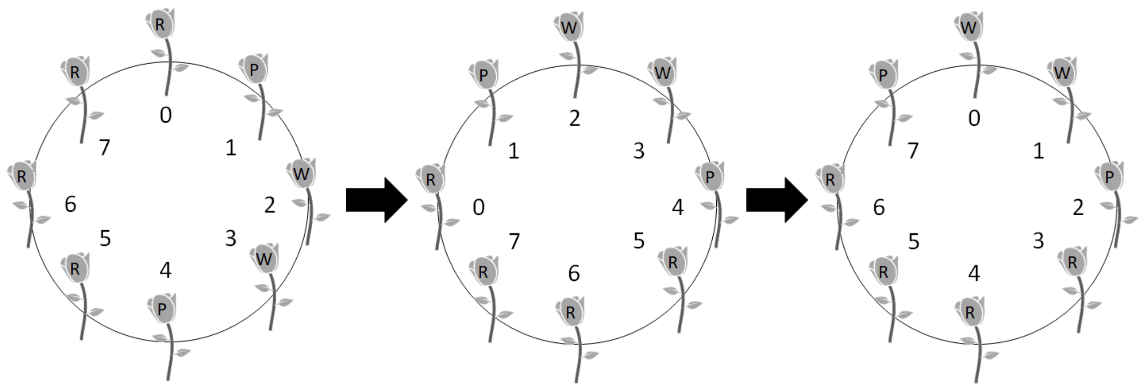
If we rotate by 2 steps (*i.e.*, `step = 2`), we will have the following bouquet:

```
("W", "W", "P", "R", "R", "R", "R", "P")
```

If we rotate by 2 steps again, we will have the following bouquet:

```
("P", "R", "R", "R", "R", "P","W", "W")
```

This can be visually represented as follows:



### Question

Write the function `rotate(bouquet, step)` to rotate the given `bouquet` (`tuple`) by `step` number of steps (`int`).

### Assumptions

- `step >= 0`

### Sample Run #1

```
>>> rotate(("R", "P", "W", "W", "P", "R", "R", "R"), 2)
('W', 'W', 'P', 'R', 'R', 'R', 'R', 'P')
```

### Sample Run #2

```
>>> rotate(("R", "P", "W", "W", "P", "R", "R", "R"), 0)
('R', 'P', 'W', 'W', 'P', 'R', 'R', 'R')
```

*continue on the next page...*

**Sample Run #3**

```
>>> rotate(("R", "P", "W", "W", "P", "R", "R", "R"), 8)
('R', 'P', 'W', 'W', 'P', 'R', 'R', 'R')
```

**Sample Run #4**

```
>>> rotate(("R", "P", "W", "W", "P", "R", "R", "R"), 7)
('R', 'R', 'P', 'W', 'W', 'P', 'R', 'R')
```

## 1.2 Iterative Flower [15 marks]

Using the function `rotate(bouquet, step)`, we can find the sequence of flowers given by Valentino more easily. The sequence of flowers should be given as a string (`str`). First, you need to find the relation between `k` and `step`.

### Question

Write the *iterative* function `flower_I(bouquet, k)` to return the sequence of flowers (`str`) given by Valentino from the circular `bouquet` (`tuple`) using the given `k` (`int`).

### Restrictions

- You may not use recursive function(s) to solve this.

### Assumptions

- `len(bouquet) >= 0` and `k > 0`

**Sample Run #1**

```
>>> flower_I(("R", "P", "W", "W", "P", "R", "R", "R"), 3)
WRRPPRWR
```

**Sample Run #2**

```
>>> flower_I(("R", "P", "W", "W", "P", "R", "R", "R"), 8)
RRWRPPRW
```

**Sample Run #3**

```
>>> flower_I(("R", "P", "W", "W", "P", "R", "R", "R"), 7)
RRRPPRWW
```

*continue on the next page...*

## 1.3 Recursive Flower [15 marks]

### Question

Write the *recursive* function `flower_R(bouquet, k)` to return the sequence of flowers (`str`) given by Valentino from the circular `bouquet` (`tuple`) using the given `k` (`int`).

### Restrictions

- You may not use iterative constructs (*e.g.*, loop, list comprehensions, *etc.*) to solve this.
- The function `flower_R` must be *recursive* (*i.e.*, it calls itself). The use of any recursive helper functions will not be counted as being recursive.

### Assumptions

- `len(bouquet) >= 0` and `k > 0`

### Sample Run #1

```
>>> flower_R(("R", "P", "W", "W", "P", "R", "R", "R"), 3)
WRRPPRWR
```

### Sample Run #2

```
>>> flower_R(("R", "P", "W", "W", "P", "R", "R", "R"), 8)
RRWRPPRW
```

### Sample Run #3

```
>>> flower_R(("R", "P", "W", "W", "P", "R", "R", "R"), 7)
RRRPPRWW
```

*continue on the next page...*

## 1.4 Pink Rose [5 marks]

Valentino wants to end with a pink rose. As a friend, you want to help him achieve his dream.

### Question

Write the function `pink_rose(bouquet)` that returns the smallest value of `k` (`int`) such that the given `bouquet` ends with a pink rose (*i.e.*, `"P"`). If no such number exists, you should return `-1` instead.

### Assumptions

- `len(bouquet) >= 0`

### Sample Run #1

```
>>> pink_rose(("R", "P", "W", "W", "P", "R", "R", "R"), 2)
12
```

## Question 2: Bouquet

To make the initial bouquet, Valentino had a small problem. All flower shops in the area sells roses as a *bundle*. The bundle consists of roses of only one colour (*e.g.*, bundle of 5 red roses, bundle of 4 white roses, *etc.*). For instance, Flowers R Us are selling the following bundles as well as their representations in this question:

- `("R", 5, 3)`: A bundle of 5 red roses costing \$3.
- `("R", 3, 2)`: A bundle of 3 red roses costing \$2.
- `("W", 4, 3)`: A bundle of 4 white roses costing \$3.
- `("W", 2, 2)`: A bundle of 2 white roses costing \$2.
- `("P", 3, 4)`: A bundle of 3 pink roses costing \$4.

To make a bouquet, Valentino must choose three bundles. However, he has some other conditions to satisfy:

- The bouquet must contain at least one pink rose.
- The bouquet must be created from at most three bundles.
- The bouquet must contain exactly the number of roses he requires.

For example:

1. A bouquet with 5 roses: `(("P", 3, 4), ("W", 2, 2))`.
   - The bundle `("P", 3, 4)` is added because it has pink rose.
   - The bouquet `(("R", 5, 3),)` and `(("R", 3, 2), ("W", 2, 2))` is not sufficient because it has no pink rose even when they have exactly 5 roses and uses fewer than three bundles.
2. A bouquet with 10 roses:
   (a) `(("P", 3, 4), ("W", 4, 3), ("R", 3, 2))`
   (b) `(("P", 3, 4), ("W", 2, 2), ("R", 5, 3))`
   (c) `(("P", 3, 4), ("W", 4, 3), ("P", 3, 4))` *repeat is allowed*

We represent a store as a sequence of bundles as above. For instance, we can have Flowers R Us represented as:
`flowers_r_us = [("R", 5, 3), ("R", 3, 2), ("W", 4, 3), ("W", 2, 2), ("P", 3, 4)]`

### 2.1 Making the Bouquet [20 marks]

#### Question

Write a function `make_bouquet(shop, number)` that check if it is possible to make the bouquet with the given `number` (`int`) of flowers from the given `shop`. Returns a Boolean (`bool`) `True` if it is possible, otherwise returns `False`.

#### Assumptions

- `number > 0`

#### Sample Run #1

```
>>> make_bouquet(flowers_r_us, 10)
True
```

**Sample Run #2**

```
>>> make_bouquet(flowers_r_us, 14)
False
```

**Note:** not enough flowers in a bundle of three containing a pink rose.

**Sample Run #3**

```
>>> make_bouquet(flowers_r_us, 2)
False
```

## 2.2 Minimum Cost [10 marks]

To safe on money, Valentino wants to find the minimum price he can find. He wants to use to remaining money to make the most beautiful Valentine's Day Card.

### Question

Write a function `minimum_cost(shop, number)` that returns the minimum cost (`int`) if it is possible to make the bouquet with the given `number` (`int`) of flowers from the given `shop`. If it is not possible to make the bouquet, returns `-1`.

### Assumptions

- `number > 0`

**Sample Run #1**

```
>>> minimum_cost(flowers_r_us, 10)
9
```

**Sample Run #2**

```
>>> minimum_cost(flowers_r_us, 14)
-1
```

**Sample Run #3**

```
>>> minimum_cost(flowers_r_us, 2)
-1
```

## Question 3: Valentine's Day Card

On the Valentine's Day itself, Valentino wants to give the last pink flower together with a DIY Valentine's Day Card. The Valentine's Day Card looks like a flower as shown below:

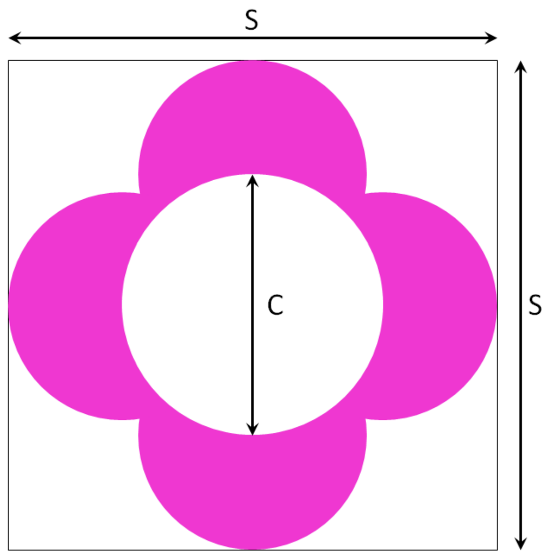


Figure 2: Valentine's Day Card with sides of length $S$ consisting of a flower with four pink petals and a centre circle.

The flower is actually made up of 6 objects listed from *bottom-to-top* as seen from above:

1. 1 × white square
   - This is the base with side of length $S$
   - We assume that the centre of the white square is at coordinate $(0, 0)$
2. 4 × pink circles
   - This made up the 4 pink petals and they are circle before being hidden by the centre circle below (*see Figure 3*)
   - The diameter of the pink circles is $D$
   - The coordinates are given as follows with respect to $S$ and $D$:
     (a) `Top`    : $(0, S/2 - D/2)$
     (b) `Right`  : $(S/2 - D/2, 0)$
     (c) `Bottom`: $(0, -S/2 + D/2)$
     (d) `Left`   : $(-S/2 + D/2, 0)$
3. 1 × white circle
   - The diameter is $C$
   - The centre of the white circle is at coordinate $(0, 0)$

With this, we can actually *parametrise* the pink flower using only three values: $S$, $C$ and $D$.

### 3.1 Get Paint [30 marks]

As a DIY project, Valentino needs to buy the paint to paint the pink area of the Valentine's Day Card. He need to calculate precisely how much money he needs to paint the card. To save on money as he wish to bring her to a nice restaurant, he *does not want to waste paint on the area covered by the centre circle*. Since the shape of the flower is irregular, it is impossible for him to calculate. He turns to you, as his best friend, to compute the area using Monte-Carlo method.

*continue on the next page...*

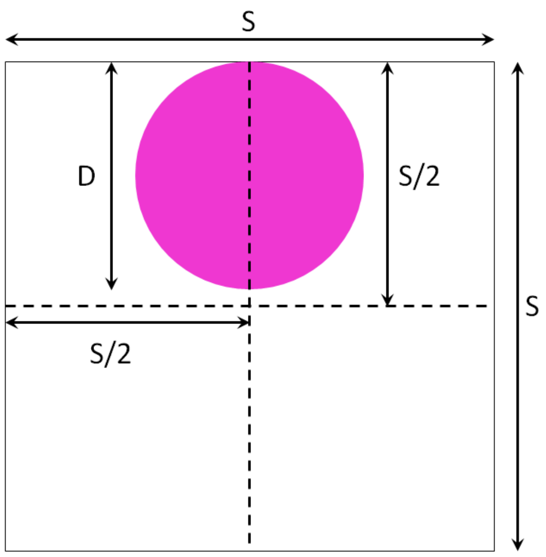


Figure 3: The top pink circle before being covered by centre circle.

The idea is simple:

1. Throw $n$ number of darts: you are supposed to use $n = 10000$.
2. If the dart falls inside any of the pink petals AND outside of the centre circle, then you count it as being inside the pink area. We let the number of darts inside pink area as $m$.
3. The $m/n$ is the ratio of pink area to the size of the card (*i.e.*, $S \times S$).

If you remember your Assignment 3, then you should know how to check if something lands inside a circle. However, this circle may not be centred at (0,0). So how do you check that?

### Question

Write the function `paint_area(S, C, D)` that takes in three floats (`float`) `S`, `C` and `D` that parametrise the Valentine's Day Card as above and computes the area of the pink coloured region using *Monte-Carlo method*.

### Assumptions

- `S > 0`, `0 < C < S`, `0 < D < S`

### Sample Run #1

```
>>> paint_area(15, 8, 7) # roughly between 86.1 to 92.3
90.25874999999999
```

### Sample Run #2

```
>>> paint_area(15, 8, 3) # roughly between 26.1 to 30.5 (but should be 9π)
28.176750000000002
```

**– End of Paper –**