# IT5001 Practical Exam

## Grading Guidelines:

- No mark will be given *if your code cannot run*, namely any syntax error or crashes.
  - Your tutors will ONLY grade what you submitted and they will **not** fix your code.
  - Hint: Comment out any part that you do not want.
- This paper has three part and each part has three tasks. And you will write one function for each task.
- Workable code that can produce correct answers will only give you *partial* marks. Only good and efficient code will give you full marks
- Marks will be deducted if your code is unnecessarily long or hard-coded.
- Please note that you cannot import extra packages or functions in some parts of this PE. But you can use all the build-in functions of Python
- You must use the same function names as those in the skeleton given
- All your function **should return the answer as output instead of printing** using "`print()`".
- There are altogether 9 Tasks in this PE. You are recommended to read through the paper first. Then, decide and strategize on your plan.

## Question 1 (30 marks)

In this Question, you **cannot import** any package or functions other than those already in the skeleton file.

### Task 1 Alphabet to Integers (10 marks)

Write a function `alpha2num(c)` to convert a character c into an integer that is its rank in alphabetical order, e.g. 'a'→ 1, 'b'→ 2, … , 'z'→26, etc. Both upper and lower case character will map to the same number. Note that your code in this Task cannot be too long. You will gain full mark in this Task **only if your code is less than 10 lines**. Sample output:

```
>>> alpha2num('a')
1
```

```
>>> alpha2num('C')
3
```

```
>>> alpha2num('R')
18
```

Hint: The alphabet song: '`abcdefghijklmnopqrstuvwxyz`'? Or, you can use the built-in function '`ord()`'.

### Task 2 String to Value (Iterative) (10 marks)

Write a function `strValueI(s)` to convert a string s into an integer that is the sum of its characters. You can assume that the string s will not contain anything other than the 26 x 2 (upper and lower cases) English alphabets. And you have to write your function by iteration without recursion. Namely, you must use meaning "`for`" or "`while`" loop(s) in this question. (List comprehensions is counted as iterations.)

```
>>> strValueI("practical")
83
```

```
>>> strValueI("is")
28
```

```
>>> strValueI("easy")
50
```

### Task 3 String to Value (Recursion) (10 marks)

Write a function `strValueR(s)` to convert a string s into the sum as Problem 2 in **recursion** without iteration. However, calling your function in previous tasks does not count as recursion in this Task.

### Submission for Question 1

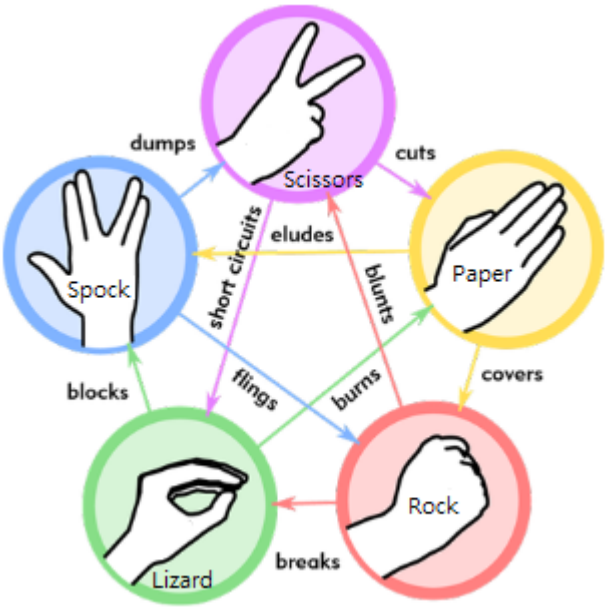
You should submit the following three functions:

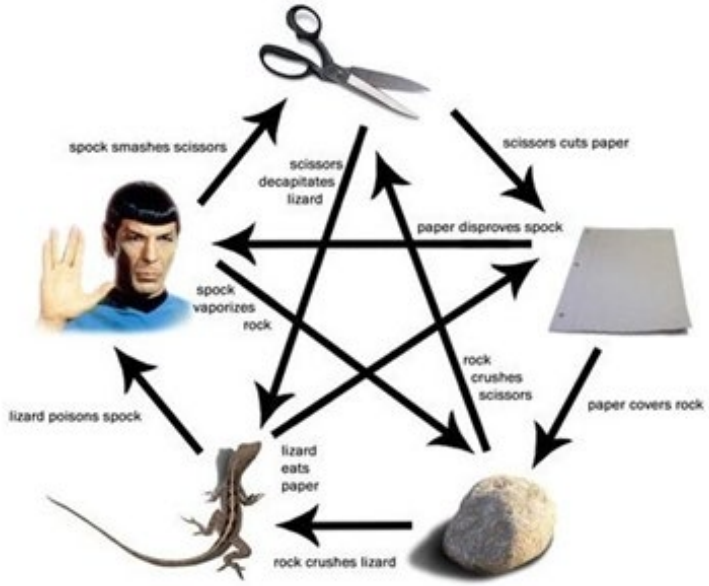
- `alpha2num(c), strValueI(s), strValueR(s)`

# Question 2 'Paper, Scissors, Rock, Lizard, Spock' (35 marks)

In this Question, you **can import** any packages or functions.

We all know the game 'paper, scissors, rock', and now we have an extended version of 'paper, scissors, rock, lizard, Spock'. The following two diagrams illustrate how to play and 'which' wins 'which'. Basically, for each of the arrows below, the starting point of the arrow wins the end of the arrow.




To simplify our implementation, we use the numbers 0 to 4 to represent the above entities. And it sums up in a dictionary given in the skeleton code:

```
psrlsdict = {0:'scissor',1:'paper',2:'rock',3:'lizard',4:'Spock'}
```

Notice that our dictionary follows the clockwise order of the above diagrams.

## Task 4 Be the Judge of the Game (10 marks)

Assume that there are two players, namely, Player 1 and Player 2. With the inputs as `p1move` and `p2move` that are two numbers that represent the moves, write a function `psrlsWhoWin(p1move,p2move)` to decide who wins. Your function will **return** the player number, either 1 or 2, if one of them wins. However, if it's a tie, return 0 instead. You assume the two inputs are always within the range from 0 to 4.

Sample output:

```
>>> print(psrlsWhoWin(2,4))
2
>>> print(psrlsWhoWin(2,3))
1
```

```
>>> print(psrlsWhoWin(4,0))
1
>>> print(psrlsWhoWin(1,1))
0
```

## Task 5 Play it N times (10 marks)

The two players decide to play a match N times. So the moves of each player is given in a list of N numbers. For example:

```
p1moves = [1,2,3,0,4,4,0]
p2moves = [2,4,1,4,3,0,2]
```

Assuming that both players will play according to the order in the sequences. Write a function `nP1Wins(p1moves,p2moves)` that will **return** the **number of wins by Player 1.** So this function will return **3** for the above inputs. You will only gain full marks if your answer in the previous Task (Task 4) is correct.

## Task 6 (Extra hard) Rearranging the Moves of Player 1 (15 marks)

After playing N times as the above. Player 1 has ***this special power*** that he can go back in time and rearrange his moves. And he can only rearrange the moves instead of changing any moves arbitrarily. If Player 1 changes his moves as:

```
[1,2,3,0,4,4,0] → [4,1,0,3,0,2,4]
```

Player 1 can win all 7 times instead of 3 in the previous game!!! Write a function `findWinningSeq(p1moves,p2moves)` to return a list of lists that includes all the sequences that can maximize the number of wins by Player 1. Here are some sample outputs.

```
>>> p1moves = [3,2,1]
>>> p2moves = [1,2,3]
>>> print(nP1Wins(p1moves,p2moves))
1
>>> newp1moves = findWinningSeq(p1moves,p2moves)
>>> print(newp1moves)
[[3, 1, 2]]
>>> print(nP1Wins(newp1moves[0],p2moves))
3
```

```
>>> p1moves = [1,2,3,0,4,4,0]
>>> p2moves = [2,4,1,4,3,0,2]
>>> print(nP1Wins(p1moves,p2moves))
3
>>> newp1moves = findWinningSeq(p1moves,p2moves)
>>> print(newp1moves)
[[4, 1, 0, 3, 0, 2, 4], [4, 3, 0, 1, 0, 2, 4]]
>>> print(nP1Wins(newp1moves[0],p2moves))
7
```

## Submission for Question 2

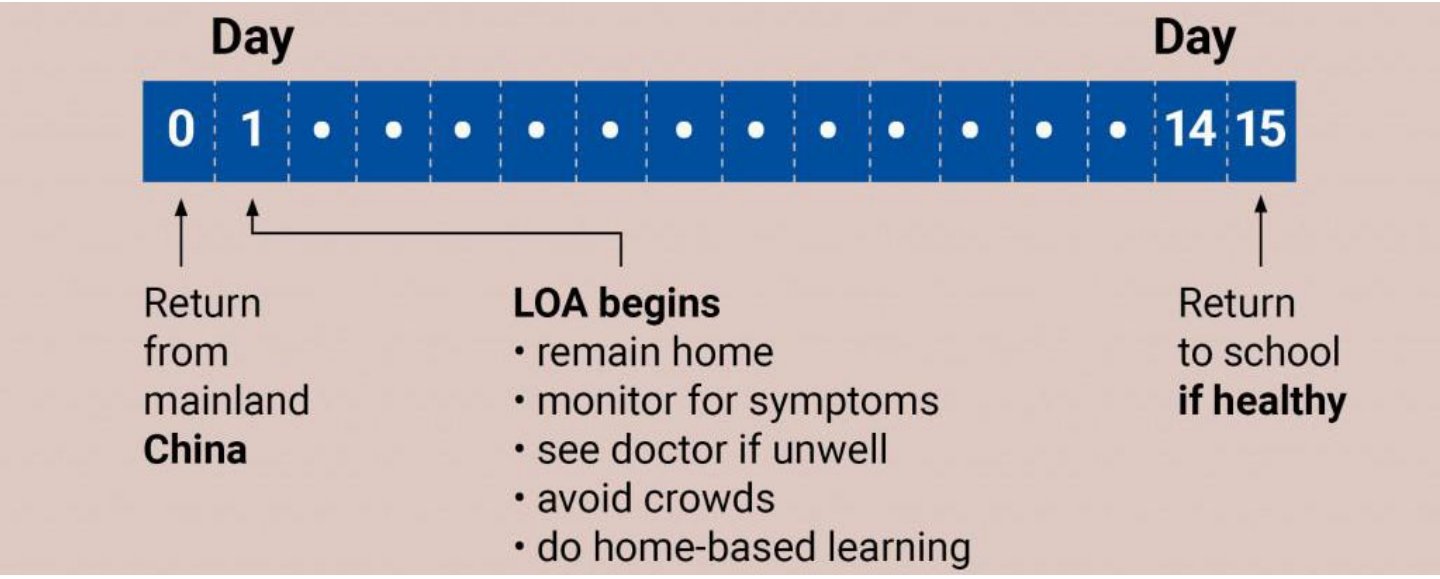
You should submit the following three functions:

- `psrlsWhoWin(p1move,p2move)`, `nP1Wins(p1moves,p2moves)`, `findWinningSeq(p1moves,p2moves)`

# Question 3 Leave of Absence (LOA) Computations (35 marks)

In this Question, you **cannot import** any package or functions other than those already in the skeleton file.

Because of the virus situation, Singapore government implemented compulsory LOA for everyone in office or school.




We will write some functions to compute these LOA data.

## Task 7 Calculating the Day of a Year (10 marks)

Given the numbers for the month and the day in a month, calculate what is the day number of that day from the beginning of the year. For example, 1st Jan is Day 1 of the year, and 31st Dec is Day 365 of the year. Write a function `doy(month,day)` to return the day number. Sample outputs:

```
>>> print(doy(1,1))
1
>>> print(doy(2,1))
32
>>> print(doy(12,31))
365
>>> print(doy(12,30))
364
```

Note that January is represented by the month number 1 (not starting from 0). We assume that February has exactly 28 days (not a leap year). And we provided a tuple for your convenience:

```
NODinAMonth = (31,28,31,30,31,30,31,31,30,31,30,31)
```

The tuple contains the number of days in each month. The item with index `i` contains the #days of (`i+1`)th month of a year.

## Task 8 List out LOA Employees (15 marks)

In order to minimize the effect of LOA, every employee will only travel to an infected country for only one day. Namely, he will fly there in the morning and return at the same night before midnight. So all the travelling records will be recorded as a tuple of tuples, e.g.

```
travelDates = (('Amy',2,14), ('Bill',2,24),('Cid',1,2),('David',2,27),
               ('Ed',1,31),('Frank',3,28),('Gary',3,11),('Ian',1,31))
```

Each record is a tuple of three data: name of the employee, the month and day of the travel.

Write a function `listLOA(td,month,day)` to compute who are the employee current in LOA on a given day and return them as a list with '`td`' being the tuple of the travel date records like the above. One final requirement, your output should be sorted in ascending order in names. And we assume no two persons have the same name in our company. For the example travel date recoreds given, here are some sample output:

```
>>> print(listLOA(travelDates,2,26))
[('Amy', 2, 14), ('Bill', 2, 24)]
>>> print(listLOA(travelDates,2,27))
[('Amy', 2, 14), ('Bill', 2, 24), ('David', 2, 27)]
>>> print(listLOA(travelDates,2,28))
[('Amy', 2, 14), ('Bill', 2, 24), ('David', 2, 27)]
>>> print(listLOA(travelDates,3,1))
[('Bill', 2, 24), ('David', 2, 27)]
```

## Task 9 Find the First Day in a Month for a Meeting with the Most People (10 marks)

The travel dates record tuple actually includes EVERY employees of the company. For each month, we want to hold a 'physical' meeting in our company that involves as many employees as possible. (Strangely that the boss does not want any video conferencing.) So given a travel date record, write a function to compute the ***first*** best date in a month to hold a meeting, that maximize the number of participants. Your function should return a tuple `(dy,nP)` such that `dy` is the first best date for the meeting with `nP` people. Sample output:

```
>>> print(firstBestMeetingDate(travelDates,2))
(15, 7)
>>> print(firstBestMeetingDate(travelDates,3))
(26, 8)
>>> print(firstBestMeetingDate(travelDates,4))
(12, 8)
```

## Submission for Question 3

You should submit the following three functions:

- `doy(month,day)`, `listLOA(travelDates,month,day)`, `firstBestMeetingDate(travelDates,month)`