CT Project 1 Report

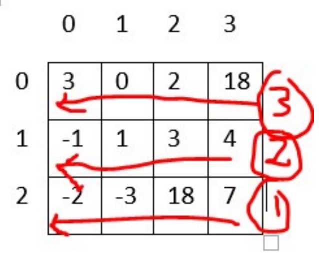
QN 1

To simulate weighted probability, a larger range of numbers will mean a higher probability of getting a random number within that range. This range with a probability is then assigned to its corresponding die number. In this example, whenever 1 to 5 is randomly generated, die number 1 will be shown. If done many times, number 1 will appear approximately 50% of the time.

Numbers from 1 to 10	Probability of getting these numbers	Corresponding die no.
1-5	0.5	1
6	0.1	2
7	0.1	3
8	0.1	4
9	0.1	5
10	0.1	6

QN 2A

The idea is to replace each value in the matrix with its SV value. But this must be done in this particular sequence so that the iterative addition works:



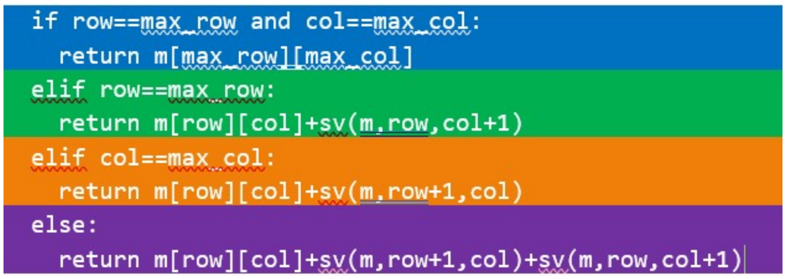
	0	1	2	3
0	3	0	2	18
1	-1	1	3	4
2	-2	-3	18	7

It starts with the cell on the bottom-most right corner and works its way leftward until the row is finished. The it starts back at the rightmost cell of the row above it. This continues all the way until row=0, col=0 is reached.

For each cell, its value plus the bottom and right values (not SVs) are added (if they exist). The total is then reassigned as the new value of the cell which is actually the SV of the cell. This method works because although we are only adding the bottom and right values, these values are now the SV values. So we are in fact adding the SV values of the right and bottom cells.

QN 2B

The main function calls another function sv which takes in the parameters m from the main function and 0, 0 to start from the leftmost, topmost value in the matrix whose row=0 and col=0.



```
 if row==max_row and col==max_col:
   return m[max_row][max_col]
 elif row==max_row:
   return m[row][col]+sv(m,row,col+1)
 elif col==max_col:
   return m[row][col]+sv(m,row+1,col)
 else:
   return m[row][col]+sv(m,row+1,col)+sv(m,row,col+1)
```

	0	1	2	3
0	🟪 3	🟪 0	🟪 2	🟧 18
1	🟪 -1	🟪 1	🟪 3	🟧 4
2	🟩 -2	🟩 -3	🟩 18	🟦 7

The function adds the value of the current cell plus the bottom and right sv values (if they exist) by recursively calling itself again with additions to the row and col parameters respectively.

The final result returned is the total addition of all the sv values broken down till the base condition is reached.

QN 3

To find 3 numbers that add up to t, I first loop through nums to get the first num, f.

A nested for loop inside gets the second possible num, s and another for loop inside this loop gets the third possible num, l. The second and third possible numbers are added up to see if they are equal to the difference between t and the first num. If equal, they are added to the new_list.

```
nums= [3, 0, 1, 0, -1, -2, 0]         # In the first iteration,

for f in range(0,len(nums)):           # first num=3
    diff=t-nums[f]                    # diff=0-3=-3
    for s in range(f+1,len(nums)):    # second num=0
      for l in range (s+1,len(nums)): # third num=1
        if nums[s]+nums[l]==diff:     # 0+1 does not equal to -3
          new_list.append([f,s,l])
  return new_list
```

Note that the inner loops' starting index is 1+ outer loop's starting index. This is to prevent duplicate answers.

A similar variation of this method is used to find 2 numbers that add up to t where the third loop to get l is removed.

Madhumitha D/O Suresh Kumar