VIKNESH ASHWIN PILLAI
SIMID: 10216853
UOWID: 7058913

CSCI262 Assignment 2

Q1

a.

Puzzle A: Puzzle B:

Expected hash (E)	Frequency (F)		Expected hash (E)	Frequency (F)
1	1		1	0
2	1		2	0
3	1		3	0
4	1		4	1
5	1		5	4
6	1		6	10
7	1		7	20
8	1		8	35
9	1		9	56
10	1		10	84
11	1		11	120
12	1		12	161
13	1		13	204
14	1		14	246
15	1		15	284
16	1		16	315
17	1		17	336
18	1		18	344
19	1		19	336
20	1		20	315
21	1		21	284
22	1		22	246
23	1		23	204
24	1		24	161
25	1		25	120
26	1		26	84
27	1		27	56
28	1		28	35
29	1		29	20
30	1		30	10
31	1		31	4
32	1		32	1

b.

I created a for loop to loop within the range of the size of the bit-string from the requirements of the puzzle and in each loop for the number of sub puzzles in the puzzle.

Puzzle A:

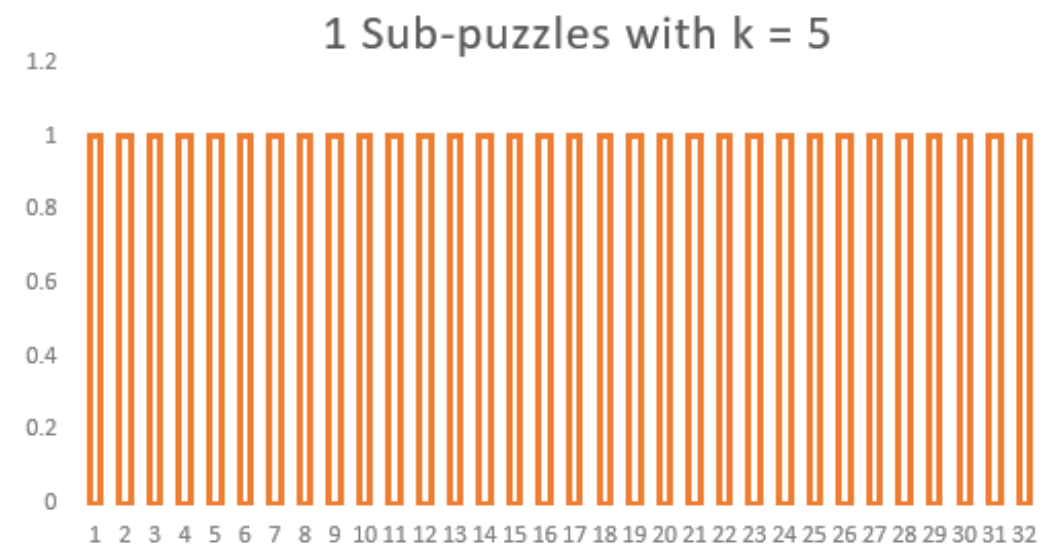
```
counter = 0
for exhash in range(1, 32+1):
    for one in range(1, 32+1):
        sum = one
        if sum == exhash:
            counter = counter + 1
    print(counter)
    counter = 0
```

Puzzle B:

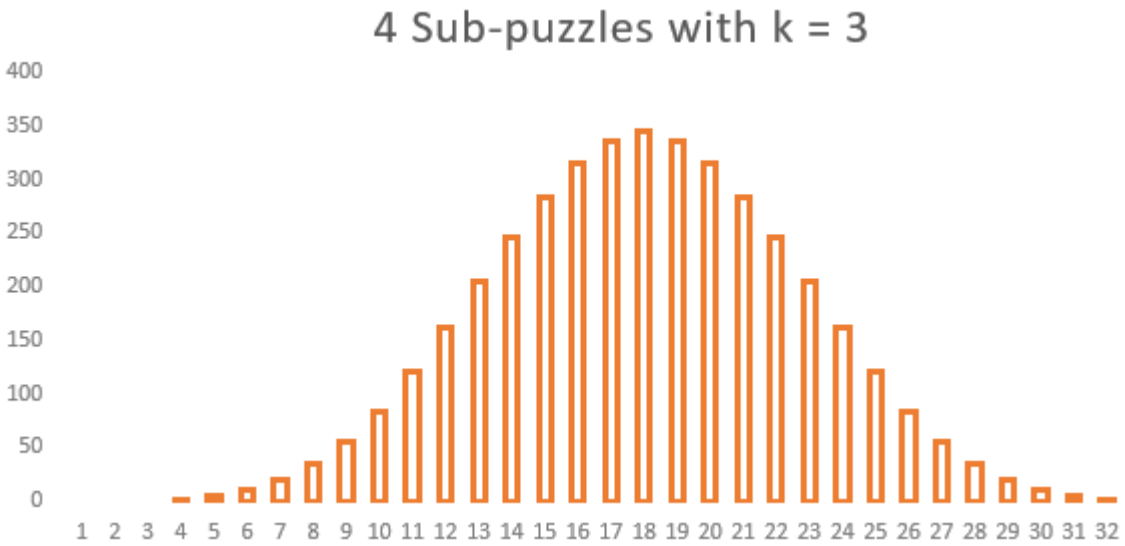
```
counter = 0
for exhash in range(1, 32 + 1):
    for one in range(1, 8 + 1):
        for two in range(1, 8 + 1):
            for three in range(1, 8 + 1):
                for four in range(1, 8 + 1):
                    sum = one + two + three + four
                    if sum == exhash:
                        counter = counter + 1
    print(counter)
    counter = 0
```

c.

Puzzle A:



Puzzle B:



d. and e.

Puzzle A

Mean:	16.5
Variance:	85.25
Standard deviation:	9.233093
	9.233093

Puzzle B

Mean:	18
Variance:	21
Standard deviation:	4.582576
	9.233093

Q2.

The original code allows the default output to grant access to the user. This fails the default deny principle of general secure coding. The code should base access decisions on permission rather than exclusion, hence allowing the default output to be an access denied message to prevent any unauthorised access to ensure security.

The amended pseudocode should be as follows:

```
permit = CheckAccess()
if (permit == Access_Granted)
    print "Access Granted"
    run Function()
else
    print "Access Denied"
```

Q3.

$P(B_1) = 0.00125 \qquad P(B_2) = 0.99875$

$P(A^C|B_1) = 0.05 \qquad P(A^C|B_2) = 0.95$

$P(A|B_1) = 0.95 \qquad P(A|B_2) = 0.05$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A^C)P(A^C)}$$

$$P(A|B) = \frac{0.05 \times 0.99875}{0.05 \times 0.99875 + 0.95 \times 0.00125}$$

$$\boldsymbol{P(A|B) = 0.97677 \approx 97.7\%}$$

Q4. Former Cisco employee purposely damaged cloud infrastructure.

A former Cisco employee, Sudhish Kasaba Ramesh, gained unauthorized access to the company's cloud infrastructure and used malicious malware to remove 456 virtual machines that were utilized by Cisco's Webex Teams service. As a result, some 16,000 Webex customers were unable to access their accounts for two weeks.

The Impact and the Malware

He entered Cisco's AWS cloud infrastructure without authorisation a few months after resigning from the tech giant and deployed malicious code from his Google Cloud Project account that erased 456 virtual machines related with the Cisco Webex Teams application, which offered video conferencing, video chat, file sharing, and other collaboration features. The malware was highly an injection script instructed to shut down and delete the files and machines on the cloud infrastructure. He further stated that in deploying the code, he behaved carelessly and wilfully ignored the significant danger that his actions may hurt Cisco.

Outcome of Cisco and Ramesh

Cisco spent roughly $1,400,000 in staff time to repair the damage to the system and reimburse over $1,000,000 to impacted customers after Webex Teams accounts were shut down for up to two weeks. However, no consumer data was compromised.

On July 13, 2020, Ramesh was charged with one count of Intentionally Accessing a Protected Computer Without Authorization and Recklessly Causing Damage. On 9th December 2020, he was sentenced to 24 months in prison. In addition to the prison sentence, he is sentenced to a one-year period of supervised release and a $15,000 fine. His sentence begins on 10th February 2021.

Q5

a. Worm X

Using the following code:

```
infected = 0
for i in range(25):
    if i == 0:
        infected = 1
    elif i > 0:
        infected *= 2
    print(f'Hrs:{i} Infected:{infected}')
```

I generated the following results placed in a table:

Hrs	PCs with X		
0	1	**12**	4096
1	2	**13**	8192
2	4	**14**	16384
3	8	**15**	32768
4	6	**16**	65536
5	32	**17**	131072
6	64	**18**	262144
7	128	**19**	524288
8	256	**20**	1048576
9	512	**21**	2097152
10	1024	**22**	4194304
11	2048	**23**	8388608

b. Worm X and Worm W

Using the following code:

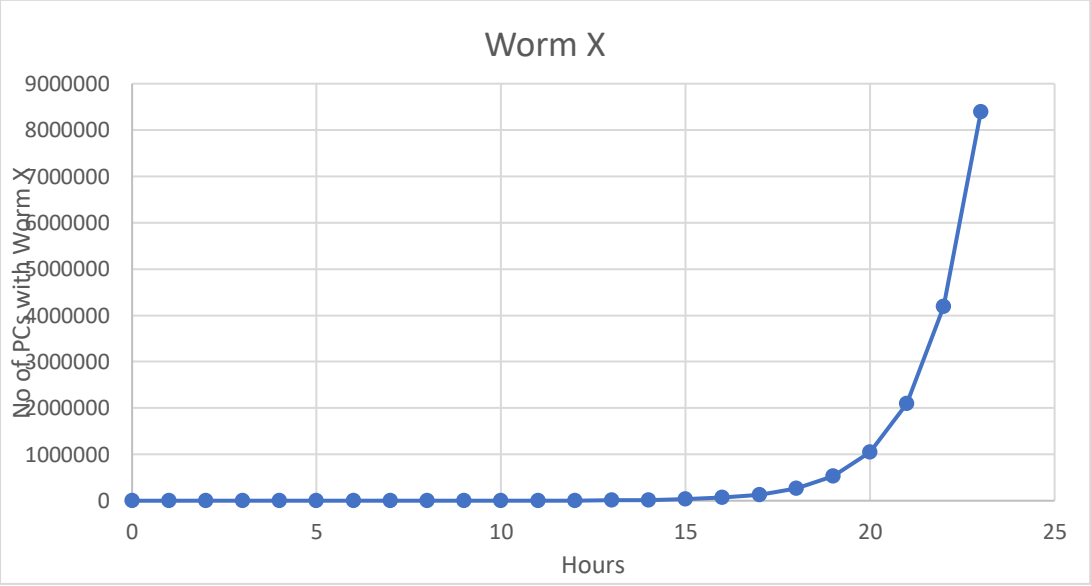
```
import numpy

infected = 0  # number of x infected computers
cured = 0  # number of w infected computers
hrs = 0

for i in numpy.arange(0, 24.5, 0.5):
    hrs = i

    if i == float(0):
        infected = 1

    if i >= float(1):
        if i % 1 == 0:  # On every hour
            infected *= 2

        if i < float(6.5):
            cured = 0  # Maintain w at hour mark

        if i % 1 == 0.5:  # on every half hour
            if i == float(6.5):  # at t = 6.5, introduce w
                cured = 1
                infected = infected - cured

            if i > float(6.5):  # after introduction of w
                cured *= 3

                if cured > infected:
                    cured = (cured / 3) + infected

                infected = int(infected - (cured - (cured / 3))
                               ) if cured < infected and infected > 0 else 0

    print(f'Hrs:{hrs} Infected:{int(infected)} Cured:{int(cured)}')
```

I generated the following results placed in a table:

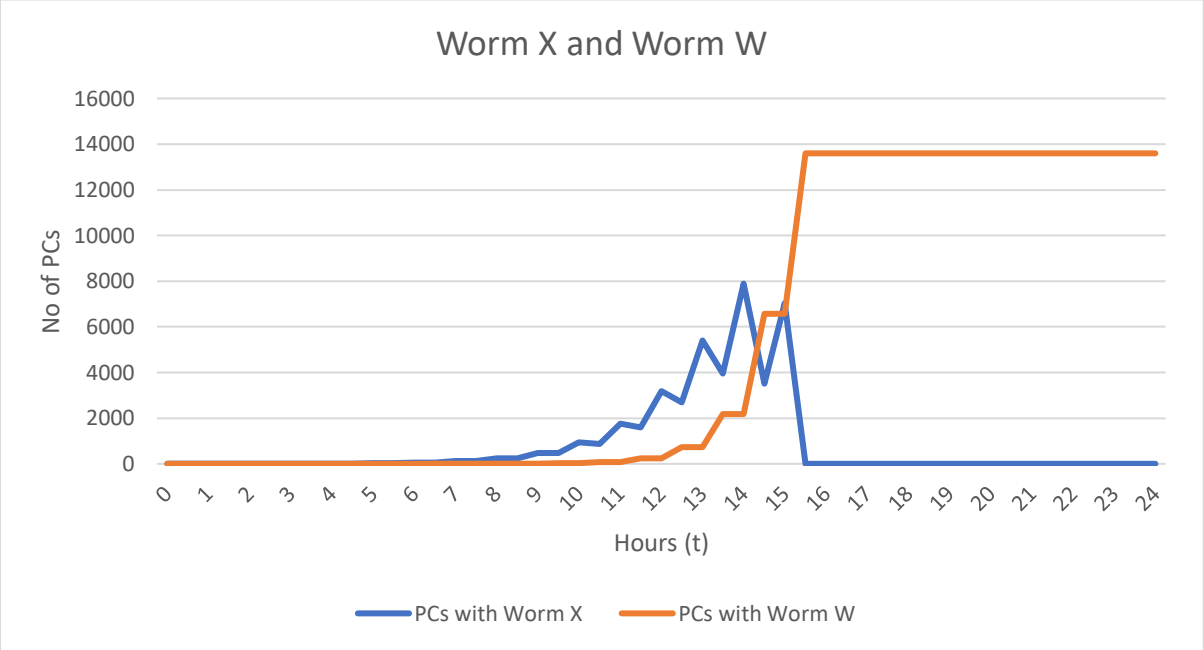
Hrs	PCs with X	PCs with W	12.0	3188	243
0.0	1	0	**12.5**	2702	729
0.5	1	0	**13.0**	5404	729
1.0	2	0	**13.5**	3946	2187
1.5	2	0	**14.0**	7892	2187
2.0	4	0	**14.5**	3518	6561
2.5	4	0	**15.0**	7036	6561
3.0	8	0	**15.5**	0	13597
3.5	8	0	**16.0**	0	13597
4.0	16	0	**16.5**	0	13597
4.5	16	0	**17.0**	0	13597
5.0	32	0	**17.5**	0	13597
5.5	32	0	**18.0**	0	13597
6.0	64	0	**18.5**	0	13597
6.5	63	1	**19.0**	0	13597
7.0	126	1	**19.5**	0	13597
7.5	124	3	**20.0**	0	13597
8.0	248	3	**20.5**	0	13597
8.5	242	9	**21.0**	0	13597
9.0	484	9	**21.5**	0	13597
9.5	466	27	**22.0**	0	13597
10.0	932	27	**22.5**	0	13597
10.5	878	81	**23.0**	0	13597
11.0	1756	81	**23.5**	0	13597
11.5	1594	243	**24.0**	0	13597

c.

Graph of Worm X



Graph with Worm X and Worm W



d.

According to the graph and table, there are already 484 infected computers at t = 9, and W has only disinfected 9 machines. If X develops, it will infect computers much more quickly, and W will be unable to keep up with the exponential growth in infection. Hence Worm W needs to have a higher infection rate to keep up with the rate of infection with Worm X. With an increased infection rate in W, in time, for this case at t = 15, it surpassed the infection rate of Worm X and eventually eradicated every trace of Worm X within 1 hr at t = 16.

Q6

a. XML Bomb

XML Bomb is a type of Denial-of-Service attack that affects a website's availability. The eXtensible Markup Language (XML) is a markup language that was created for the purpose of storing and transmitting structured data. Before being processed, data files might be examined by a parser library. The validation and comparison criteria for the type of data that occurs in the XML file are then performed using XML schemas and Document Type Definitions (DTDs) files. When the parser library is not configured properly, inline DTDs can be misused, resulting in an XML bomb, also known as an Entity Expansion XML bomb. The "Billions Laugh Effect" is a well-known XML bomb example. This form of attack can list entity definitions in a densely layered XML file, resulting in parsing and extremely huge files. Because these processed XML files are so huge, they might cause a server to crash.

b. BlueSmack

BlueSmack is a type of DoS attack that targets Bluetooth-enabled devices and the Bluetooth protocol specifically. It's comparable to the Windows 95's 'Ping of Death' exploit, in which a l2ping data packet (approximately 600 bytes) is sent to a Bluetooth device with a larger maximum packet size. This renders the gadget inoperable and has the potential to deplete the device's battery. The l2ping packet is included in the BlueZ utility package that comes standard with Linux.

c. Mydoom

Mydoom is a type of computer worm that was found in 2004 and distributed by email, affecting Microsoft Windows PCs. It attacked PCs by copying itself to shared folders on P2P KaZaA clients, and its payload included a backdoor Remote Access component on TCP port 3127. The worm was able to gather email addresses from the system and then deliver itself as an attachment from the host email server using SMTP protocols. Mydoom's original edition is the fastest-growing spam email worm ever discovered, and it has also been found to avoid sending itself to particular domain addresses, including Rutgers, MIT, Stanford, UC Berkeley, Microsoft, and Symantec. Other worm variants were then created to employ infected hosts as zombies in a Distributed Denial-of-Service assault against the SCO Group and Microsoft.

d. Torpig

Torpig is a botnet that targets Windows-based computers. It attacks users by masquerading as a Trojan horse and installing the Mebroot rootkit via a drive-by-download server. The Mebroot rootkit changes the infected system's Master Boot Record (MBR), which subsequently installs the rootkit when the system is rebooted. The virus updates its modules by communicating with a Command and Control (C&C) server through HTTP via an encrypted protocol. Computers that have been infected become members of the botnet. Torpig is a type of spyware that compromises the integrity and security of systems by examining data from impacted applications (such as Service Control Managers, email clients, the file manager, and others) in order to collect online account and password details.