Question 1

a)

We have:

$$P(A, b_1) = \emptyset_1(A,B) \times \emptyset_4(D,A) \times \emptyset_3(C,D) \times \emptyset_2(B,C)$$

Assigning B = b^1, then $\emptyset_1(A,B)$ becomes $\emptyset_5(A)$ and $\emptyset_2(B,C)$ becomes $\emptyset_6(C)$:

A	$\emptyset_5(A)$
a^0	5
a^1	10

C	$\emptyset_6(C)$
c^0	1
c^1	100

Now we have:

$$P(A, b_1) = \emptyset_5(A) \times \emptyset_4(D,A) \times \emptyset_3(C,D) \times \emptyset_6(C)$$

Applying elimination ordering: C,D, we have:

$$P(A, b_1) = \sum_{C,D} \emptyset_5(A) \times \emptyset_4(D,A) \times \emptyset_3(C,D) \times \emptyset_6(C)$$
$$= \emptyset_5(A) \sum_D \emptyset_4(D,A) \sum_C \emptyset_3(C,D) \times \emptyset_6(C)$$

Eliminating C:

$\emptyset_3(C,D) \times \emptyset_6(C)$:

C	D	Val
c^0	d^0	1×1=1
c^0	d^1	100×1 = 100
c^1	d^0	100×100 = 10000
c^1	d^1	1×100=100

Summing out C to get $\emptyset_7(D)$:

D	$\emptyset_7(D)$
d^0	1+10000= 10001
d^1	100+100= 200

Now we have:

$$P(A, b_1) = \sum_{C,D} \emptyset_5(A) \times \emptyset_4(D,A) \times \emptyset_3(C,D) \times \emptyset_6(C)$$
$$= \emptyset_5(A) \sum_D \emptyset_4(D,A) \times \emptyset_7(D)$$

Eliminating D:

$\emptyset_4(D, A) \times \emptyset_7(D)$:

D	A	Val
d^0	a^0	100×10001 = 1000100
d^0	a^1	1×10001 = 10001
d^1	a^0	1×200= 200
d^1	a^1	100×200 = 20000

Summing out D to get $\emptyset_8(A)$:

A	$\emptyset_8(A)$
a^0	1000100+200 = 1000300
a^1	10001+20000 = 30001

Now we have:

$$P(A, b_1) = \emptyset_5(A) \times \emptyset_8(A)$$

$\emptyset_5(A) \times \emptyset_8(A)$:

A	$\emptyset_9(A)$
a^0	5×1000300 = 5001500
a^1	10×30001 = 300010

Normalizing:

Since $\sum_{a \in dom(A)} \emptyset_9(A)$ = 5001500+300010= 5301510, we have:

A	$P(A \mid b_1)$
a^0	$\frac{5001500}{5301510} = 0.94341$
a^1	$\frac{300010}{5301510} = 0.05659$

Part b

In [287…

```
import random
from random import choice
from matplotlib import pyplot as plt
```

In [288…

```
def get_f_AB(A,B):
    if A==0:
        if B == 0:
            return 30
        elif B == 1:
            return 5
    elif A == 1:
        if B == 0:
            return 1
        elif B == 1:
            return 10
```

In [289…

```
def get_f_DA(D,A):
    if D==0:
        if A == 0:
            return 100
        elif A == 1:
            return 1
    elif D == 1:
        if A == 0:
            return 1
        elif A == 1:
            return 100
```

In [290…

```python
def get_f_BC(B,C):
    if B==0:
        if C == 0:
            return 100
        elif C == 1:
            return 1
    elif B == 1:
        if C == 0:
            return 1
        elif C == 1:
            return 100
```

In [291…

```python
def get_f_CD(C,D):
    if C==0:
        if D == 0:
            return 1
        elif D == 1:
            return 100
    elif C == 1:
        if D == 0:
            return 100
        elif D == 1:
            return 1
```

In [292…

```python
def normalize(a,b):
    total = a + b
    return (a/total, b/total)

def resample(probOfZeroVal):
    # 1: true, 0: false
    # |---0val---|-------1val------|
    # 0                            1
    return random.uniform(0,1) > probOfZeroVal
```

In [293...

```
def generate_sample(curr_sample_var, A,C,D):
    B = 1  # observed value
    if curr_sample_var == "A":
        (a0, a1) = (get_f_AB(0,B)* get_f_DA(D, 0), get_f_AB(1,B)* get_f_DA(D, 1))
        (a0, a1) = normalize(a0, a1)
        A = resample(a0)
    elif curr_sample_var == "C":
        (c0, c1) = (get_f_BC(B,0)*get_f_CD(0,D), get_f_BC(B,1)*get_f_CD(1,D))
        (c0, c1) = normalize(c0, c1)
        C = resample(c0)
    elif curr_sample_var == "D":
        (d0, d1) = (get_f_CD(C, 0)* get_f_DA(0, A), get_f_CD(C, 1)* get_f_DA(1, A))
        (d0, d1) = normalize(d0, d1)
        D = resample(d0)
    return A,C,D
```

In [294...

```
def next_sample_var(curr_var):
    return choice(["A", "C", "D"])
```

In [295...

```
a0_num = 0
a1_num = 0
sample_size = 1000000
curr_sample_var = "A"
B = 1  # fixed, observed value

# randomly initialize values for A,C,D
A = choice([0, 1])
C = choice([0, 1])
D = choice([0, 1])


# probA[0] = probability of a0 given b1
# probA[1] = probability of a1 given b1
probA = [[],[]]
```

In [296...

```
for i in range(1, sample_size + 1):
    if A == 0:
        a0_num += 1
    elif A == 1:
        a1_num += 1
    probA[0].append(a0_num/i)
    probA[1].append(a1_num/i)

    # select next variable to sample on
    curr_sample_var = next_sample_var(curr_sample_var)
    (A, C, D) = generate_sample(curr_sample_var, A,C,D)
```

In [309...

```
GS_a0 = probA[0][sample_size-1]
print("The estimate of P(A=a0|b1) using Gibbs Sampling with {} samples is: {:.5}"
        .format(sample_size, GS_a0))
```

```
The estimate of P(A=a0|b1) using Gibbs Sampling with 1000000 samples is: 0.94343
```

In [307...

```
GS_a1 = probA[1][sample_size-1]
print("The estimate of P(A=a1|b1) using Gibbs Sampling with {} samples is: {:.5}"
        .format(sample_size,GS_a1))
```

```
The estimate of P(A=a1|b1) using Gibbs Sampling with 1000000 samples is: 0.056574
```

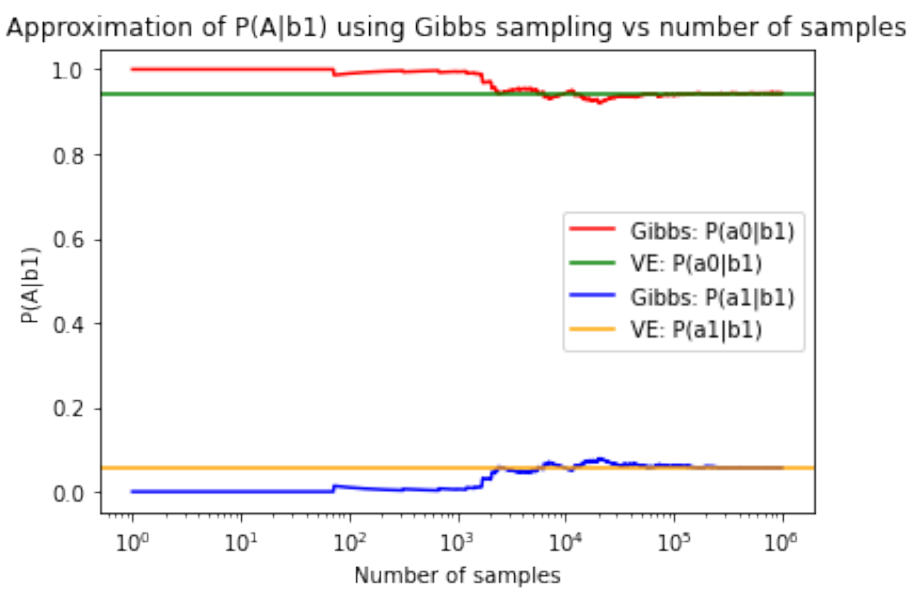
In [308…

```
x = list(range(1, sample_size+1))
y0 = probA[0]
y1 = probA[1]

# P(a0|b1) from Question 1 part a
ve_a0 = 5001500/5301510

# P(a1|b1) from Question 1 part a
ve_a1 = 300010/5301510

plt.plot(x,y0, c = "r")
plt.axhline(ve_a0, c = "g")
plt.plot(x,y1, c = "b")
plt.axhline(ve_a1, c = "orange")
plt.legend(labels = ["Gibbs: P(a0|b1)","VE: P(a0|b1)",
                     "Gibbs: P(a1|b1)", "VE: P(a1|b1)"])
plt.title('Approximation of P(A|b1) using Gibbs sampling vs number of samples')
plt.xlabel('Number of samples')
plt.ylabel('P(A|b1)')
plt.xscale("log")
plt.show()
```



Question 2

a)

Applying Resolution, we have α = ~(A ↔ C) and the KB is:

1. (B ∨ C) ∧ ~A
2. ~(B→A)
3. A→(~B ∨ C)
4. A ∨ E
5. C

6. ~(~B ∨ A)	by eliminating implication, clause 2
7. ~~B ∧ ~A	by de Morgan's rule, clause 6
8. B ∧ ~A	by double negation, clause 7
9. ~A ∨ ~B ∨ C	by eliminating implication, clause 3

Converting α into CNF:

10. ~(A ↔ C)	
11. ~((A→C) ∧ (C→A))	by eliminating biconditional, clause 10
12. ~((~A∨C) ∧ (~C∨A))	by eliminating implication, clause 11
13. ~(~A∨C) ∨ ~(~C∨A)	by de Morgan's rule, clause 12
14. (~~A∧~C) ∨ (~~C∧~A)	by de Morgan's rule, clause 13
15. (A∧~C) ∨ (C∧~A)	by double negation, clause 14

Hence, ~α is:

16. ~((A∧~C) ∨ (C∧~A))	by negation, clause 15
17. ~(A∧~C) ∧ ~ (C∧~A)	by de Morgan's rule, clause 16
18. (~A∨~~C) ∧ (~C∨~~A)	by de Morgan's rule, clause 17
19. (~A∨C) ∧ (~C∨A)	by double negation, clause 18

Converting all the clauses to CNF, we have:

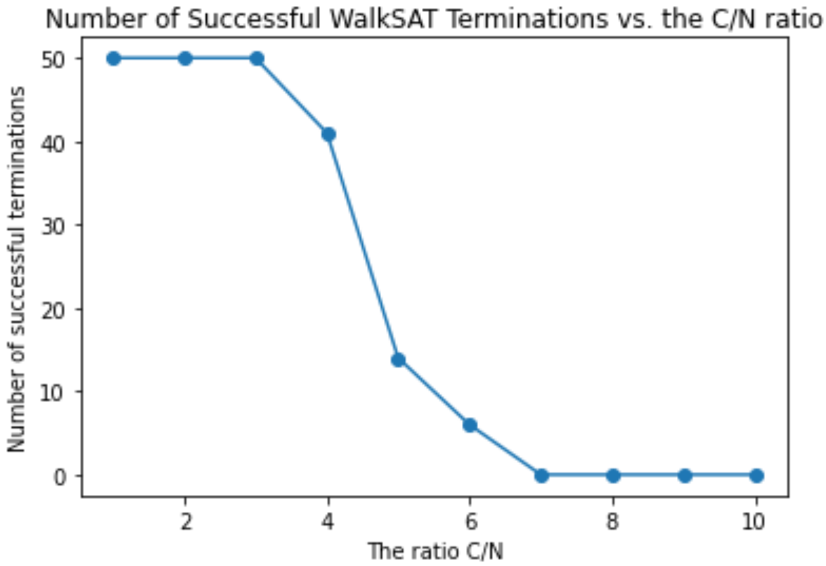
1. B ∨ C
2. ~A
3. A ∨ E
4. C
5. B
6. ~A ∨ ~B ∨ C
7. ~A∨C
8. ~C∨A

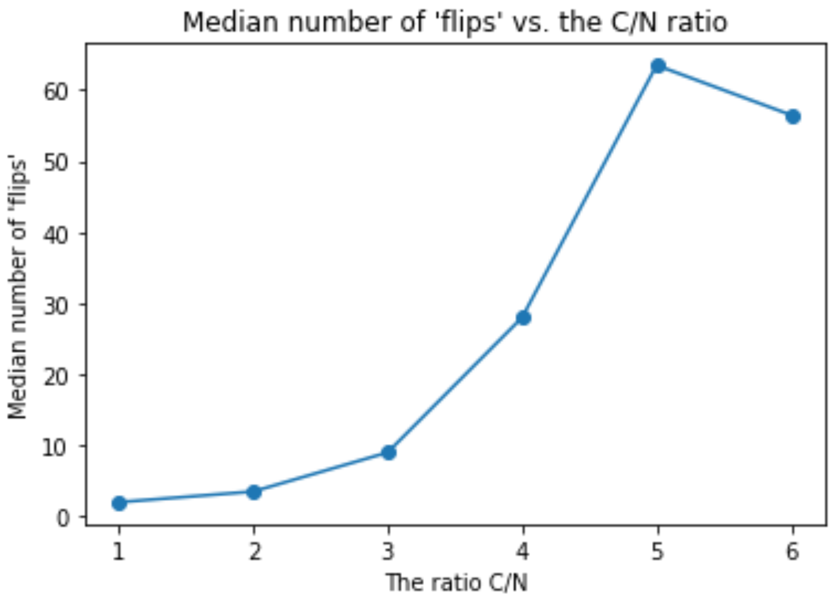
- resolve 1 and 6, we have:
  9. ~A ∨ C
- resolve 1 and 8, we have:
  10. B ∨ A
- resolve 2 and 8, we have:
  11. ~C

- resolve 4 and 11, we have empty clause: {}
  This indicates contradiction; thus KB$\wedge$~α is unsatisfiable and therefore, the query is entailed (i.e., KB $\models$ ~(A $\leftrightarrow$ C)).

(b) According to the answer from part a, KB entails ~(A $\leftrightarrow$ C). Hence, there is no case (i.e., no interpretation) in which KB is true and ~(A $\leftrightarrow$ C) is false.

Question 3
Site used: https://gitlab.com/HenryKautz/Walksat

- 

- 

Question 4

- 10 objects and 10 constant symbols:
  Each object is assigned with 1 symbol, and hence, there are 10! interpretations.

- 2 ternary predicates:
  For each predicate, there are three arguments, thus having $10\times10\times 10= 1000$ combinations. Each combination can either be true or false, so there are 2^{1000} interpretations. Hence, for 2 ternary predicates, there are $(2^{1000})^2 = 2^{2000}$ interpretations.

- 2 binary predicates:
  For each predicate, there are two arguments, thus having $10\times10= 100$ combinations. Each combination can either be true or false, so there are 2^{100} interpretations. Hence, with 2 binary predicates, we have $(2^{100})^2= 2^{200}$ interpretations.

- 10 unary predicates:
  For each predicate, there are one argument, thus having 10 combinations. Each combination can either be true or false, so there are 2^{10} interpretations. With 10 unary predicates, there are $(2^{10})^{10}= 2^{100}$ interpretations.

Therefore, there are $10! \times 2^{2000} \times 2^{200} \times 2^{100} = 10! \times 2^{2300}$ interpretations in total.