1. Generate truth tables for following compound propositions.

a. (p ∧ ⌐q v r) → (⌐p v r)

b. (p ∧ ⌐q v r) ↔ (⌐p v r)

Source Code:

```
#include<stdio.h>
#include<stdbool.h>
int main(){

    bool p[] = {0, 0, 0, 0, 1, 1, 1, 1};
    bool q[] = {0, 0, 1, 1, 0, 0, 1, 1};
    bool r[] = {0, 1, 0, 1, 0, 1, 0, 1};
    printf("a. (p ∧ ⌐q v r) → (⌐p v r) \n");
    printf(" ____________________________________________________________________
    __________________________\n");
    printf("|   p   |   q   |   r   |   ⌐p  |   ⌐q  | (p ∧ ⌐q v r) | (⌐p v r) |
     (p ∧ ⌐q v r) → (⌐p v r)  |\n");
    printf("|_______|_______|_______|_______|_______|_______________|__________
    |___________________________|\n");

    for (int i = 0; i < 8; i++)
    {
        bool a = p[i] && !q[i] || r[i];
        bool b = !p[i] || r[i];
        bool c;
        if( b == 1)             c = 1;
        else if( a == 0 )       c = 1;
        else                    c = 0;

        printf("|   %d   |   %d   |   %d   |   %d   |   %d   |      %d      |    %d
        |            %d          |\n", p[i], q[i], r[i], !p[i], !q[i], a, b, c);
    }

    printf("|________|_______|_______|_______|_______|_______________|__________|
    __________________________|\n");
```

```
    printf("\n\nb. (p ∧ ⌐q ∨ r) ↔ (⌐p ∨ r)\n");
    printf(" _____________________________________________________________
    _____________________________\n");
    printf("|   p   |   q   |   r   |   ⌐p  |   ⌐q  | (p ∧ ⌐q ∨ r) | (⌐p ∨ r) |
    (p ∧ ⌐q ∨ r) ↔ (⌐p ∨ r)  |\n");
    printf("|_______|_______|_______|_______|_______|_______________|__________|
    _____________________________|\n");

    for (int i = 0; i < 8; i++)
    {
        bool a = p[i] && !q[i] || r[i];
        bool b = !p[i] || r[i];
        bool c;
        if ((b == 1 && a == 1) || (b == 0 && a == 0))          c = 1;
        else                                                   c = 0;

        printf("|   %d   |   %d   |   %d   |   %d   |   %d   |      %d      |    %d
        |            %d         |\n", p[i], q[i], r[i], !p[i], !q[i], a, b, c);
    }

printf("|________|_______|_______|_______|_______|_______________|__________|
_____________________________|\n");
}
```

Output:

```
➜ Lab 3 git:(master) ✗ gcc src/01.c -o bin/01
➜ Lab 3 git:(master) ✗ ./bin/01
a. (p ∧ ¬q ∨ r) → (¬p ∨ r)
 ---------------------------------------------------------------------------------------------
|   p   |   q   |   r   |   ¬p   |   ¬q   | (p ∧ ¬q ∨ r) | (¬p ∨ r) | (p ∧ ¬q ∨ r) → (¬p ∨ r)  |
|-------|-------|-------|--------|--------|--------------|----------|-------------------------|
|   0   |   0   |   0   |   1    |   1    |       0      |     1    |            1            |
|   0   |   0   |   1   |   1    |   1    |       1      |     1    |            1            |
|   0   |   1   |   0   |   1    |   0    |       0      |     1    |            1            |
|   0   |   1   |   1   |   1    |   0    |       1      |     1    |            1            |
|   1   |   0   |   0   |   0    |   1    |       1      |     0    |            0            |
|   1   |   0   |   1   |   0    |   1    |       1      |     1    |            1            |
|   1   |   1   |   0   |   0    |   0    |       0      |     0    |            1            |
|   1   |   1   |   1   |   0    |   0    |       1      |     1    |            1            |
|-------|-------|-------|--------|--------|--------------|----------|-------------------------|


b. (p ∧ ¬q ∨ r) ↔ (¬p ∨ r)
 ---------------------------------------------------------------------------------------------
|   p   |   q   |   r   |   ¬p   |   ¬q   | (p ∧ ¬q ∨ r) | (¬p ∨ r) | (p ∧ ¬q ∨ r) ↔ (¬p ∨ r)  |
|-------|-------|-------|--------|--------|--------------|----------|-------------------------|
|   0   |   0   |   0   |   1    |   1    |       0      |     1    |            0            |
|   0   |   0   |   1   |   1    |   1    |       1      |     1    |            1            |
|   0   |   1   |   0   |   1    |   0    |       0      |     1    |            0            |
|   0   |   1   |   1   |   1    |   0    |       1      |     1    |            1            |
|   1   |   0   |   0   |   0    |   1    |       1      |     0    |            0            |
|   1   |   0   |   1   |   0    |   1    |       1      |     1    |            1            |
|   1   |   1   |   0   |   0    |   0    |       0      |     0    |            1            |
|   1   |   1   |   1   |   0    |   0    |       1      |     1    |            1            |
|-------|-------|-------|--------|--------|--------------|----------|-------------------------|
➜ Lab 3 git:(master) ✗
```

2. Use a truth table to test the validity of the following argument using truth table.

If you are a hound dog, then you howl at the moon.

You don't howl at the moon.

Therefore, you aren't a hound dog.

Source Code:

```
#include<stdio.h>
#include<stdbool.h>

bool ifThen(bool a, bool b){
    if( b == 1)             return 1;
    else if( a == 0 )       return 1;
    else                    return 0;
}
int main (){

    printf(" The argument is as: \n\
        If you are a hound dog, then you howl at the moon.\n\
        You don't howl at the moon.\n\
        Therefore, you aren't a hound dog.\n");

    printf("\n\
        Let, p = you are a hound dog\n\
            q = you howl at the moon.\n");

    printf("According to argument\n\
        p → q\n\
        ¬q \n\
        -------\n\
        ∴ ¬p\n\n");

    bool p[] = {0, 0, 1, 1};
    bool q[] = {0, 1, 0, 1};

    printf("\t\t\t _____________________________________\n");
    printf("\t\t\t|   p   |   q   | (p → q) | ¬q  |    ¬p  |\n");
    printf("\t\t\t|_______|_______|_________|_____|________|\n");
```

```
    for (int i = 0; i < 4; i++)
    {
        bool a = ifThen(p[i], q[i]);
        bool b = !q[i];
        bool c = !p[i];
        printf("\t\t\t|   %d   |   %d   |    %d    | %d  |    %d   |\n", p[i], q[i],
        a,b,c);
    }
        printf("\t\t\t|_______|_______|_________|_____|________|\n");
}
```

Output:

```
➜  Lab 3 git:(master) ✗ gcc src/02.c -o bin/02
➜  Lab 3 git:(master) ✗ ./bin/02
 The argument is as:
        If you are a hound dog, then you howl at the moon.
        You don't howl at the moon.
        Therefore, you aren't a hound dog.

        Let, p = you are a hound dog
             q = you howl at the moon.
According to argument
        p → q
        ⌐q
        -------
        ∴ ⌐p

                        _________________________________________
                       |   p   |   q   | (p → q) | ⌐q  |    ⌐p   |
                       |_______|_______|_________|_____|_________|
                       |   0   |   0   |    1    |  1  |    1    |
                       |   0   |   1   |    1    |  0  |    1    |
                       |   1   |   0   |    0    |  1  |    0    |
                       |   1   |   1   |    1    |  0  |    0    |
                       |_______|_______|_________|_____|_________|
➜  Lab 3 git:(master) ✗
```

3. Test the validity of the following argument using truth table.

I will buy a new goat or a used Yugo.

If I buy both a new goat and a used Yugo, I will need a loan.

I bought a used Yugo and I don't need a loan.

Therefore, I didn't buy a new goat.

Source Code:

```
#include<stdio.h>
#include<stdbool.h>
bool ifThen(bool a, bool b){
    if( b == 1)                 return 1;
       else if( a == 0 )        return 1;
       else                     return 0;
}

int main(){
    printf(" The argument is as: \n\
       I will buy a new goat or a used Yugo.\n\
       If I buy both a new goat and a used Yugo, I will need a loan.\n\
       I bought a used Yugo and I don't need a loan.\n\
       Therefore, I didn't buy a new goat.\n");

    printf("\n\
       Let, p = I will buy a new goat\n\
           q = I will buy a used Yugo\n\
           r = I will need a loan\n");

    printf("According to argument\n\
       p ∨ q \n\
       (p ∧ q) → r\n\
       q ∧ ¬r\n\
       -------------\n\
       ∴ ¬p\n\n");

    bool p[] = {0, 0, 0, 0, 1, 1, 1, 1};
    bool q[] = {0, 0, 1, 1, 0, 0, 1, 1};
    bool r[] = {0, 1, 0, 1, 0, 1, 0, 1};
```

```
    printf("\t\t\t _____________________________________________________
    _____________\n");
    printf("\t\t\t|   p   |   q   |   r   |   ⌐r  | p ∨ q  | (p ∧ q) → r |
     (q ∧ ⌐r)  |   ⌐p  |\n");
    printf("\t\t\t|_______|_______|_______|_______|________|_____________|
    ____________|_______|\n");

    for (int i = 0; i < 8; i++)
    {
        bool a = p[i] || q[i];
        bool b = ifThen((p[i] && q[i]), r[i]);
        bool c = q[i] && !r[i];
        bool d = !p[i];

        printf("\t\t\t|   %d   |   %d   |   %d   |   %d   |    %d   |
       %d    |    %d    |   %d   |\n", p[i], q[i], r[i], !r[i], a, b, c, d);
    }
        printf("\t\t\t|_______|_______|_______|_______|________|_____________|
        ____________|_______|\n");
}
```

Output:

```
➜  Lab 3 git:(master) ✗ gcc src/03.c -o bin/03
➜  Lab 3 git:(master) ✗ ./bin/03
 The argument is as:
        I will buy a new goat or a used Yugo.
        If I buy both a new goat and a used Yugo, I will need a loan.
        I bought a used Yugo and I don't need a loan.
        Therefore, I didn't buy a new goat.

        Let, p = I will buy a new goat
             q = I will buy a used Yugo
             r = I will need a loan
According to argument
        p ∨ q
        (p ∧ q) → r
        q ∧ ⌐r
        -------------
        ∴ ⌐p

                     _____________________________________________________________________________
                    |   p   |   q   |   r   |   ⌐r  | p ∨ q  | (p ∧ q) → r | (q ∧ ⌐r)  |   ⌐p  |
                    |_______|_______|_______|_______|________|_____________|___________|_______|
                    |   0   |   0   |   0   |   1   |    0   |       1     |     0     |   1   |
                    |   0   |   0   |   1   |   0   |    0   |       1     |     0     |   1   |
                    |   0   |   1   |   0   |   1   |    1   |       1     |     1     |   1   |
                    |   0   |   1   |   1   |   0   |    1   |       1     |     0     |   1   |
                    |   1   |   0   |   0   |   1   |    1   |       1     |     0     |   0   |
                    |   1   |   0   |   1   |   0   |    1   |       1     |     0     |   0   |
                    |   1   |   1   |   0   |   1   |    1   |       0     |     1     |   0   |
                    |   1   |   1   |   1   |   0   |    1   |       1     |     0     |   0   |
                    |_______|_______|_______|_______|________|_____________|___________|_______|
➜  Lab 3 git:(master) ✗
```

4. Test the validity of the following argument using truth table.

Premise: If I go to the mall, I will buy new jeans

Premise: If I buy new jeans, I will buy a shirt to go with it.

Conclusion: If I go to the mall, I will buy a shirt.

Source Code:

```
#include<stdio.h>
#include<stdbool.h>

bool ifThen(bool a, bool b){
    if( b == 1)             return 1;
    else if( a == 0 )       return 1;
    else                    return 0;
}

int main(){

    printf(" The argument is as: \n\
        Premise: If I go to the mall, I will buy new jeans\n\
        Premise: If I buy new jeans, I will buy a shirt to go with it.\n\
        Conclusion: If I go to the mall, I will buy a shirt.\n");

    printf("\n\
        Let, p = I go to the mall\n\
            q = I will buy new jeans\n\
            r = I will buy a shirt\n");

    printf("According to argument\n\
         p → q\n\
         q → r\n\
       ---------\n\
        ∴ p → r\n");

    bool p[] = {0, 0, 0, 0, 1, 1, 1, 1};
    bool q[] = {0, 0, 1, 1, 0, 0, 1, 1};
    bool r[] = {0, 1, 0, 1, 0, 1, 0, 1};

    printf("\t\t\t _____\n");
```

```
    printf("\t\t\t|   p   |   q   |   r   | p → q | q → r | p → r |\n");
    printf("\t\t\t|_______|_______|_______|_______|_______|_______|\n");

    for (int i = 0; i < 8; i++)
    {
        bool a = ifThen(p[i], q[i]);
        bool b = ifThen(q[i], r[i]);
        bool c = ifThen(p[i], r[i]);
        printf("\t\t\t|   %d   |   %d   |   %d   |   %d   |   %d   |   %d   |\n",
        p[i], q[i], r[i], a, b, c);
    }
    printf("\t\t\t|_______|_______|_______|_______|_______|_______|\n");
}
```

Output:

```
➜  Lab 3 git:(master) ✗ gcc src/04.c -o bin/04
➜  Lab 3 git:(master) ✗ ./bin/04
 The argument is as:
        Premise: If I go to the mall, I will buy new jeans
        Premise: If I buy new jeans, I will buy a shirt to go with it.
        Conclusion: If I go to the mall, I will buy a shirt.

        Let, p = I go to the mall
             q = I will buy new jeans
             r = I will buy a shirt
According to argument
         p → q
         q → r
        ---------
         ∴ p → r
                        _____________________________________________
                       |   p   |   q   |   r   | p → q | q → r | p → r |
                       |_______|_______|_______|_______|_______|_______|
                       |   0   |   0   |   0   |   1   |   1   |   1   |
                       |   0   |   0   |   1   |   1   |   1   |   1   |
                       |   0   |   1   |   0   |   1   |   0   |   1   |
                       |   0   |   1   |   1   |   1   |   1   |   1   |
                       |   1   |   0   |   0   |   0   |   1   |   0   |
                       |   1   |   0   |   1   |   0   |   1   |   1   |
                       |   1   |   1   |   0   |   1   |   0   |   0   |
                       |   1   |   1   |   1   |   1   |   1   |   1   |
                       |_______|_______|_______|_______|_______|_______|
➜  Lab 3 git:(master) ✗
```