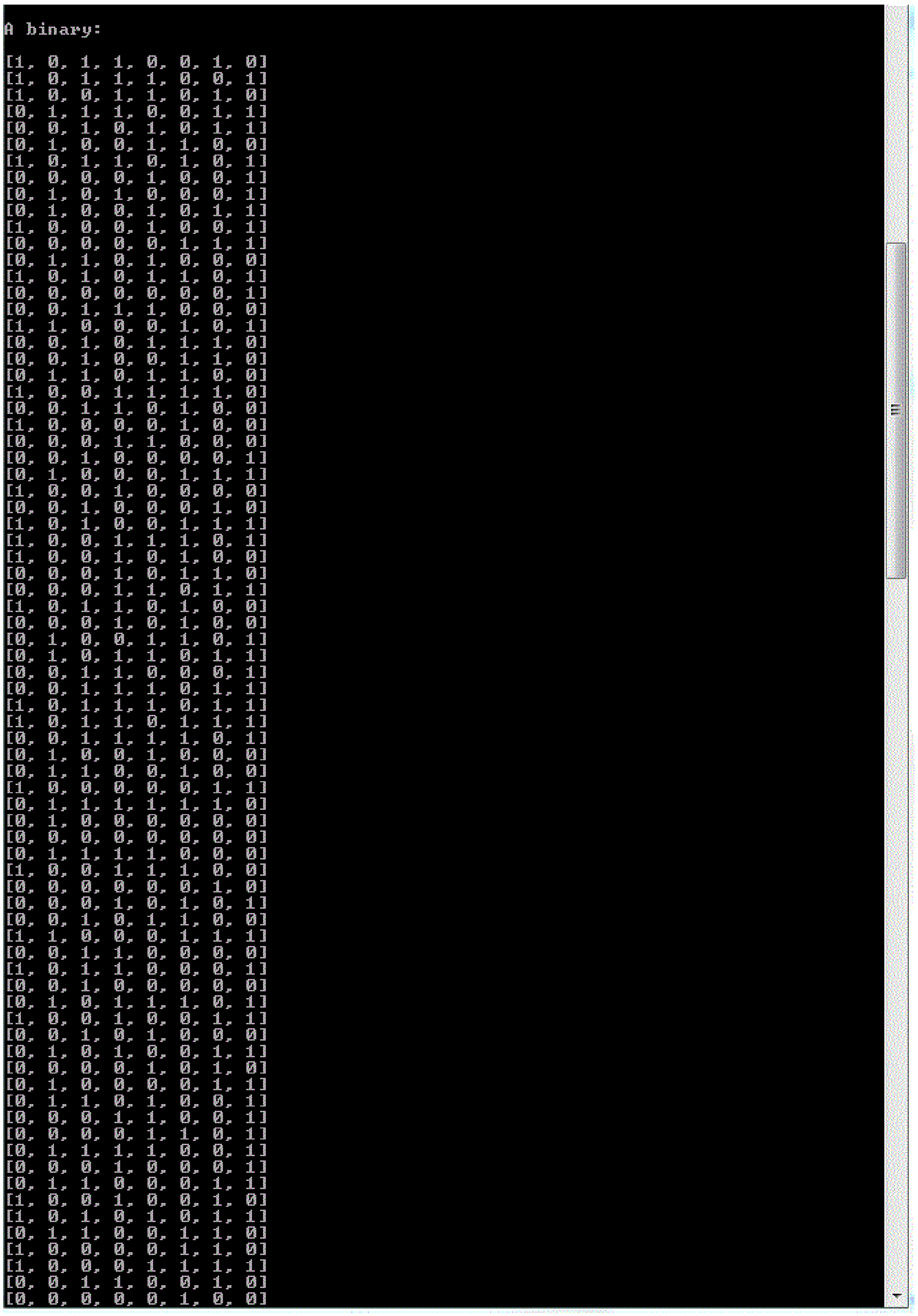
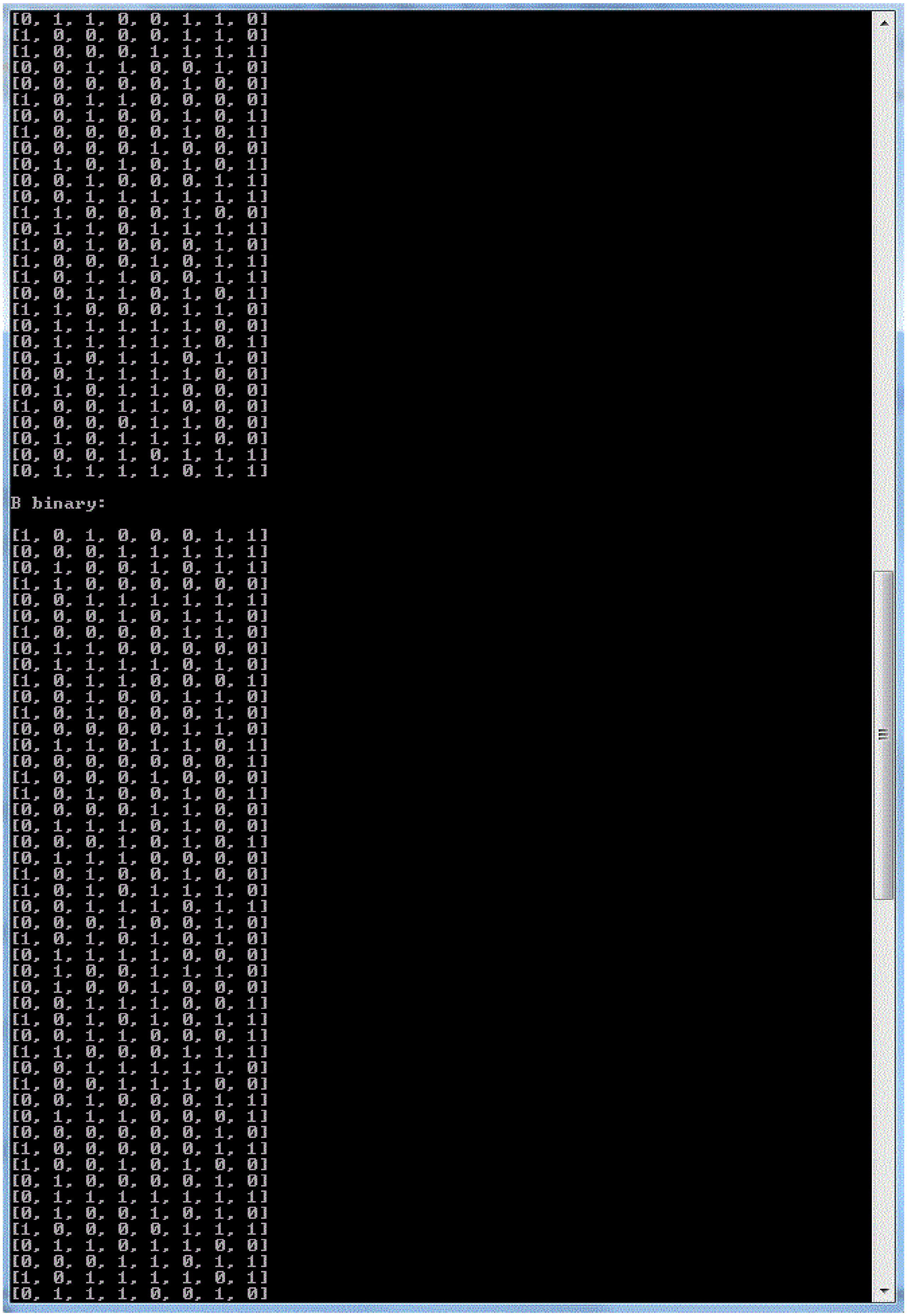
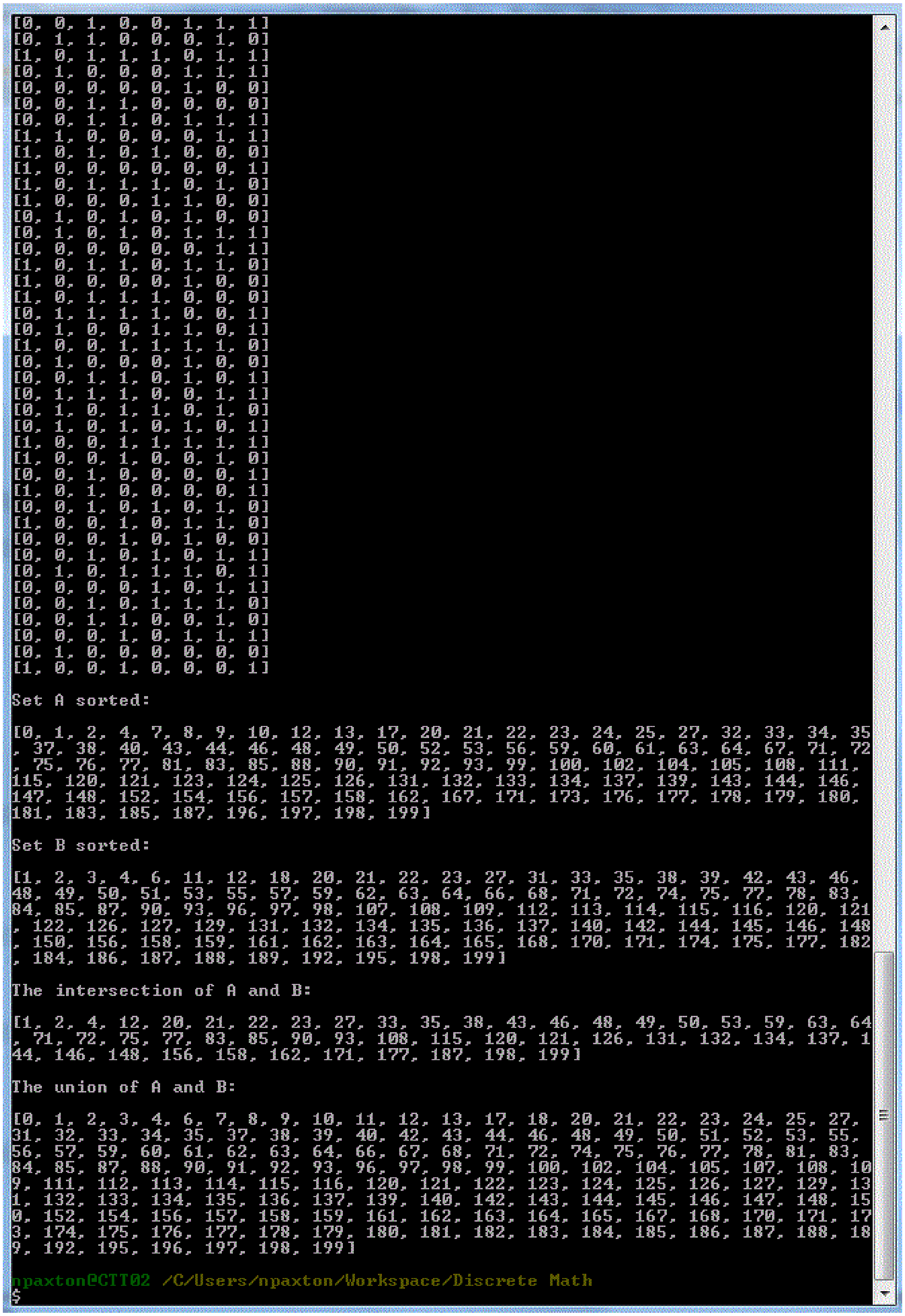
```python
"""
The purpose of this program is to build to arrays of 100 numbers each.
The numbers for each array will be random numbers between 1 and 200.

The arrays will be displayed in their binary representations and the
binary arrays will be compared to find the union and the intersection.

Once the union and intersection are found, the binary will be converted
back to decimal.  The output will show the intersection and union, as
well as both input sets sorted.

The outputs are sorted using the native sort functions in Python.  The
outputs will all be sorted in ascending order.
"""
import random

class projectFive:

    def __init__(self):

        #Global variables
        self.aData = []
        self.bData = []

        #Instantiate the arrays
        A = self.irand(100, 200)
        B = self.irand(100, 200)

        #Convert the A data to binary
        for i in range(100):
            self.aData.append(self.bin(A[i], 8))

        #Print the A binary set
        print("\nA binary: \n")
        for i in range(100):
            print(self.aData[i])

        #Convert the B data to binary
        for i in range(100):
            self.bData.append(self.bin(B[i], 8))

        #Print the B binary set
        print("\nB binary: \n")
        for i in range(100):
            print(self.bData[i])

        #Print the sorted decimal sets
        A.sort()
        B.sort()
        print("\nSet A sorted: \n")
        print(A)
        print("\nSet B sorted: \n")
        print(B)

        self.abIntersect(A, B)
        self.abUnion(A, B)

    def irand(self, n, m):
        b = list(range(n))
        b = random.sample(range(m), n)
        return b

    def bin(self, n, m):
        a = []
        while n > 0:

            #Check for even numbers
            if n % 2 == 0:
                n = int(n/2)
                a.append(0)

            #Check for odds
            elif n % 2 == 1:
                n = int(n/2)
                a.append(1)

        #Check current length against specified length and add zeroes
        while len(a) < m:
            a.append(0)

        return list(reversed(a))

    def abIntersect(self, a, b):
        intersection = []

        #Nested for loops will handle comparison
        for i in range(len(a)):
            for j in range(len(b)):
                if a[i] == b[j]:
                    intersection.append(a[i])

        print("\nThe intersection of A and B: \n")
        intersection.sort()
        print(intersection)

    def abUnion(self, a, b):
        union = []

        for i in range(len(a)):
            union.append(a[i])

        for i in range(len(b)):
            union.append(b[i])

        #Remove the duplicates from the union
        union = list(set(union))
        union.sort()
        print("\nThe union of A and B: \n")
        print(union)

p = projectFive()
```



```
A binary:

[1, 0, 1, 1, 0, 0, 1, 0]
[1, 0, 1, 1, 1, 0, 0, 1]
[1, 0, 0, 1, 1, 0, 1, 0]
[0, 1, 1, 1, 0, 0, 1, 1]
[0, 0, 1, 0, 1, 0, 1, 1]
[0, 1, 0, 0, 1, 1, 0, 0]
[1, 0, 1, 1, 0, 1, 0, 1]
[0, 0, 0, 0, 1, 0, 0, 1]
[0, 1, 0, 1, 0, 0, 0, 1]
[0, 1, 0, 0, 1, 0, 1, 1]
[1, 0, 0, 0, 1, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 1, 1]
[0, 1, 1, 0, 1, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 1, 1, 1, 0, 0, 0]
[1, 1, 0, 0, 0, 1, 0, 1]
[0, 0, 1, 0, 1, 1, 1, 0]
[0, 0, 1, 0, 0, 1, 1, 0]
[0, 1, 1, 0, 1, 1, 0, 0]
[1, 0, 0, 1, 1, 1, 1, 0]
[0, 0, 1, 1, 0, 1, 0, 0]
[1, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 1]
[0, 1, 0, 0, 0, 1, 1, 1]
[1, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 1, 0]
[1, 0, 1, 0, 0, 1, 1, 1]
[1, 0, 0, 1, 1, 1, 0, 1]
[1, 0, 0, 1, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 1, 1, 0]
[0, 0, 0, 1, 1, 0, 1, 1]
[1, 0, 1, 1, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 1, 0, 0]
[0, 1, 0, 0, 1, 1, 0, 1]
[0, 1, 0, 1, 1, 0, 1, 1]
[0, 0, 1, 1, 0, 0, 0, 1]
[0, 0, 1, 1, 1, 0, 1, 1]
[1, 0, 1, 1, 1, 0, 1, 1]
[1, 0, 1, 1, 0, 1, 1, 1]
[0, 0, 1, 1, 1, 1, 0, 1]
[0, 1, 0, 0, 1, 0, 0, 0]
[0, 1, 1, 0, 0, 1, 0, 0]
[1, 0, 0, 0, 0, 0, 1, 1]
[0, 1, 1, 1, 1, 1, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 1, 1, 0, 0, 0]
[1, 0, 0, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 1]
[0, 0, 1, 0, 1, 1, 0, 0]
[1, 1, 0, 0, 0, 1, 1, 1]
[0, 0, 1, 1, 0, 0, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 1]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 1, 0, 1, 1, 1, 0, 1]
[1, 0, 0, 1, 0, 0, 1, 1]
[0, 0, 1, 0, 1, 0, 0, 0]
[0, 1, 0, 1, 0, 0, 1, 1]
[0, 0, 0, 0, 1, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 1, 1]
[0, 1, 1, 0, 1, 0, 0, 1]
[0, 0, 0, 1, 1, 0, 0, 1]
[0, 0, 0, 0, 1, 1, 0, 1]
[0, 1, 1, 1, 1, 0, 0, 1]
[0, 0, 0, 1, 0, 0, 0, 1]
[0, 1, 1, 0, 0, 0, 1, 1]
[1, 0, 0, 1, 0, 0, 1, 0]
[1, 0, 1, 0, 1, 0, 1, 1]
[0, 1, 1, 0, 0, 1, 1, 0]
[1, 0, 0, 0, 0, 1, 1, 0]
[1, 0, 0, 0, 1, 1, 1, 1]
[0, 0, 1, 1, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 0, 0]
```



```
[0, 1, 1, 0, 0, 1, 1, 0]
[1, 0, 0, 0, 0, 1, 1, 0]
[1, 0, 0, 0, 1, 1, 1, 1]
[0, 0, 1, 1, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 0, 1]
[1, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 1, 0, 1, 0, 1, 0, 1]
[0, 0, 1, 0, 0, 0, 1, 1]
[0, 0, 1, 1, 1, 1, 1, 1]
[1, 1, 0, 0, 0, 1, 0, 0]
[0, 1, 1, 0, 1, 1, 1, 1]
[1, 0, 1, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0, 1, 1]
[1, 0, 1, 1, 0, 0, 1, 1]
[0, 0, 1, 1, 0, 1, 0, 1]
[1, 1, 0, 0, 0, 1, 1, 0]
[0, 1, 1, 1, 1, 1, 0, 0]
[0, 1, 1, 1, 1, 1, 0, 1]
[0, 1, 0, 1, 1, 0, 1, 0]
[0, 0, 1, 1, 1, 1, 0, 0]
[0, 1, 0, 1, 1, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 1, 0, 0]
[0, 1, 0, 1, 1, 1, 0, 0]
[0, 0, 0, 1, 0, 1, 1, 1]
[0, 1, 1, 1, 1, 0, 1, 1]

B binary:

[1, 0, 1, 0, 0, 0, 1, 1]
[0, 0, 0, 1, 1, 1, 1, 1]
[0, 1, 0, 0, 1, 0, 1, 1]
[1, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1]
[0, 0, 0, 1, 0, 1, 1, 0]
[1, 0, 0, 0, 0, 1, 1, 0]
[0, 1, 1, 0, 0, 0, 0, 0]
[0, 1, 1, 1, 1, 0, 1, 0]
[1, 0, 1, 1, 0, 0, 0, 1]
[0, 0, 1, 0, 0, 1, 1, 0]
[1, 0, 1, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 1, 0]
[0, 1, 1, 0, 1, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 1]
[1, 0, 0, 0, 1, 0, 0, 0]
[1, 0, 1, 0, 0, 1, 0, 1]
[0, 0, 0, 0, 1, 1, 0, 0]
[0, 1, 1, 1, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 1, 0, 1]
[0, 1, 1, 1, 0, 0, 0, 0]
[1, 0, 1, 0, 0, 1, 0, 0]
[1, 0, 1, 0, 1, 1, 1, 0]
[0, 0, 1, 1, 1, 0, 1, 1]
[0, 0, 0, 1, 0, 0, 1, 0]
[1, 0, 1, 0, 1, 0, 1, 0]
[0, 1, 1, 1, 1, 0, 0, 0]
[0, 1, 0, 0, 1, 1, 1, 0]
[0, 1, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 1, 1, 0, 0, 1]
[1, 0, 1, 0, 1, 0, 1, 1]
[0, 0, 1, 1, 0, 0, 0, 1]
[1, 1, 0, 0, 0, 1, 1, 1]
[0, 0, 1, 1, 1, 1, 1, 0]
[1, 0, 0, 1, 1, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 1, 1]
[0, 1, 1, 1, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 0, 0, 1, 1]
[1, 0, 0, 1, 0, 1, 0, 0]
[0, 1, 0, 0, 0, 0, 1, 0]
[0, 1, 1, 1, 1, 1, 1, 1]
[0, 1, 0, 0, 1, 0, 1, 0]
[1, 0, 0, 0, 0, 1, 1, 1]
[0, 1, 1, 0, 1, 1, 0, 0]
[0, 0, 0, 1, 1, 0, 1, 1]
[1, 0, 1, 1, 1, 1, 0, 1]
[0, 1, 1, 1, 0, 0, 1, 0]
```



```
[0, 0, 1, 0, 0, 1, 1, 1]
[0, 1, 1, 0, 0, 0, 1, 0]
[1, 0, 1, 1, 1, 0, 1, 1]
[0, 1, 0, 0, 0, 1, 1, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 1, 1, 1]
[1, 1, 0, 0, 0, 0, 1, 1]
[1, 0, 1, 0, 1, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 1]
[1, 0, 1, 1, 1, 0, 1, 0]
[1, 0, 0, 0, 1, 1, 0, 0]
[0, 1, 0, 1, 0, 1, 0, 0]
[0, 1, 0, 1, 0, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 1, 1]
[1, 0, 1, 1, 0, 1, 1, 0]
[1, 0, 0, 0, 0, 1, 0, 0]
[1, 0, 1, 1, 1, 0, 0, 0]
[0, 1, 1, 1, 1, 0, 0, 1]
[0, 1, 0, 0, 1, 1, 0, 1]
[1, 0, 0, 1, 1, 1, 1, 0]
[0, 1, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 1, 0, 1, 0, 1]
[0, 1, 1, 1, 0, 0, 1, 1]
[0, 1, 0, 1, 1, 0, 1, 0]
[0, 1, 0, 1, 0, 1, 0, 1]
[1, 0, 0, 1, 1, 1, 1, 1]
[1, 0, 0, 1, 0, 0, 1, 0]
[0, 0, 1, 0, 0, 0, 0, 1]
[1, 0, 1, 0, 0, 0, 0, 1]
[0, 0, 1, 0, 1, 0, 1, 0]
[1, 0, 0, 1, 0, 1, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 0]
[0, 0, 1, 0, 1, 0, 1, 1]
[0, 1, 0, 1, 1, 1, 0, 1]
[0, 0, 0, 0, 1, 0, 1, 1]
[0, 0, 1, 0, 1, 1, 1, 0]
[0, 0, 1, 1, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1, 1, 1]
[0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 0, 0, 1]

Set A sorted:

[0, 1, 2, 4, 7, 8, 9, 10, 12, 13, 17, 20, 21, 22, 23, 24, 25, 27, 32, 33, 34, 35
, 37, 38, 40, 43, 44, 46, 48, 49, 50, 52, 53, 56, 59, 60, 61, 63, 64, 67, 71, 72
, 75, 76, 77, 81, 83, 85, 88, 90, 91, 92, 93, 99, 100, 102, 104, 105, 108, 111,
115, 120, 121, 123, 124, 125, 126, 131, 132, 133, 134, 137, 139, 143, 144, 146,
147, 148, 152, 154, 156, 157, 158, 162, 167, 171, 173, 176, 177, 178, 179, 180,
181, 183, 185, 187, 196, 197, 198, 199]

Set B sorted:

[1, 2, 3, 4, 6, 11, 12, 18, 20, 21, 22, 23, 27, 31, 33, 35, 38, 39, 42, 43, 46,
48, 49, 50, 51, 53, 55, 57, 59, 62, 63, 64, 66, 68, 71, 72, 74, 75, 77, 78, 83,
84, 85, 87, 90, 93, 96, 97, 98, 107, 108, 109, 112, 113, 114, 115, 116, 120, 121
, 122, 126, 127, 129, 131, 132, 134, 135, 136, 137, 140, 142, 144, 145, 146, 148
, 150, 156, 158, 159, 161, 162, 163, 164, 165, 168, 170, 171, 174, 175, 177, 182
, 184, 186, 187, 188, 189, 192, 195, 198, 199]

The intersection of A and B:

[1, 2, 4, 12, 20, 21, 22, 23, 27, 33, 35, 38, 43, 46, 48, 49, 50, 53, 59, 63, 64
, 71, 72, 75, 77, 83, 85, 90, 93, 108, 115, 120, 121, 126, 131, 132, 134, 137, 1
44, 146, 148, 156, 158, 162, 171, 177, 187, 198, 199]

The union of A and B:

[0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 17, 18, 20, 21, 22, 23, 24, 25, 27,
31, 32, 33, 34, 35, 37, 38, 39, 40, 42, 43, 44, 46, 48, 49, 50, 51, 52, 53, 55,
56, 57, 59, 60, 61, 62, 63, 64, 66, 67, 68, 71, 72, 74, 75, 76, 77, 78, 81, 83,
84, 85, 87, 88, 90, 91, 92, 93, 96, 97, 98, 99, 100, 102, 104, 105, 107, 108, 10
9, 111, 112, 113, 114, 115, 116, 120, 121, 122, 123, 124, 125, 126, 127, 129, 13
1, 132, 133, 134, 135, 136, 137, 139, 140, 142, 143, 144, 145, 146, 147, 148, 15
0, 152, 154, 156, 157, 158, 159, 161, 162, 163, 164, 165, 167, 168, 170, 171, 17
3, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 18
9, 192, 195, 196, 197, 198, 199]

npaxton@CTT02 /C/Users/npaxton/Workspace/Discrete Math
$
```