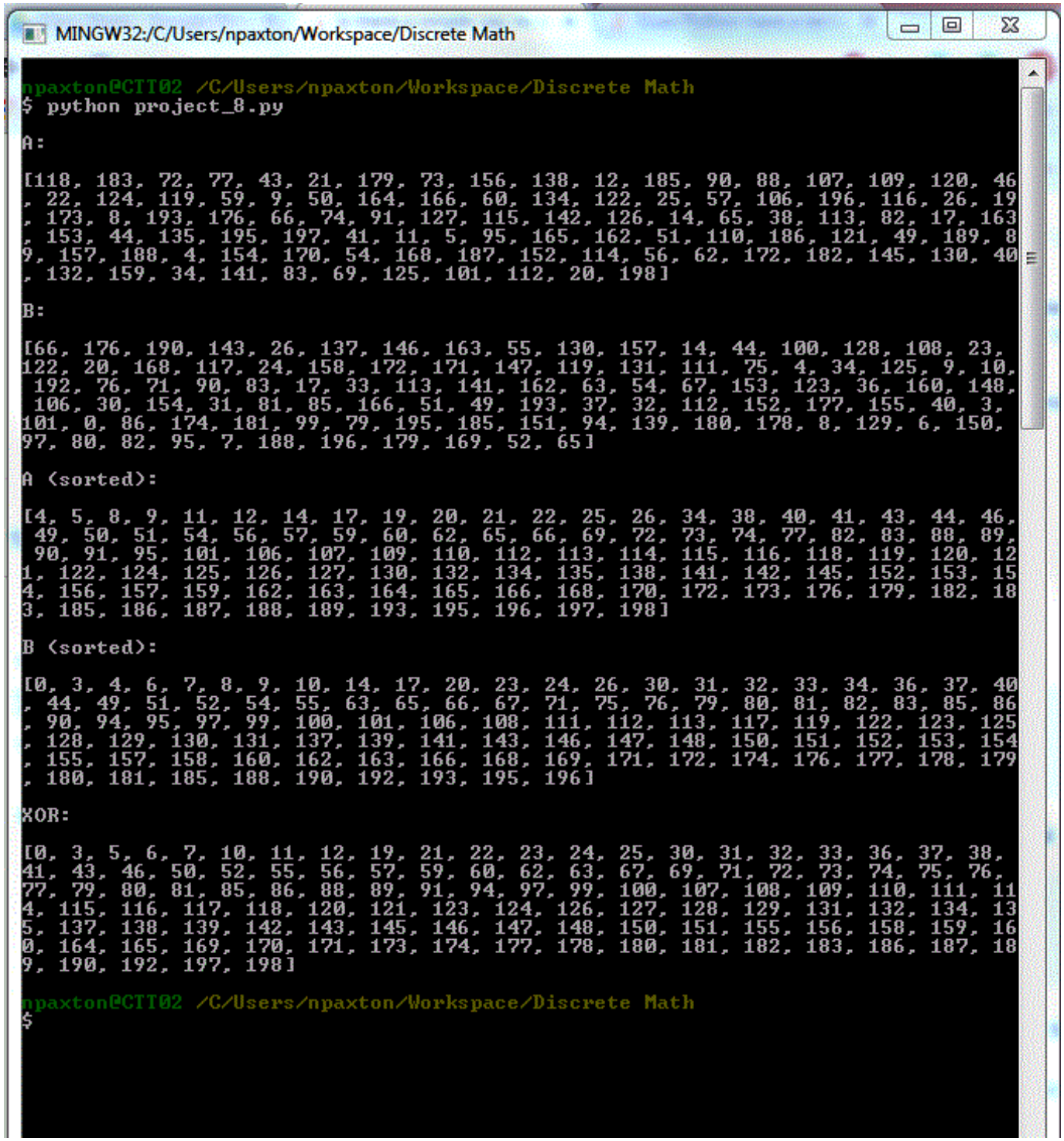
```
"""
The purpose of this program is to perform the XOR sort on two lists
of 100 numbers between 1 and 200.

The lists will be generated by using the irand function from project 5.
The binary search written for project 6 will be modified as necessary to
sort the values that are only in one list or the other.

The lists will be combined and a method will iterate over the combined
list, checking to see if the value at position [i] is equal to the value
at position [i + 1].  If there is a match, both will be popped from the
final XOR list.

"""

import random

class XOR:

    def __init__(self):
        a = self.irand(100, 200)
        b = self.irand(100, 200)

        print('\nA:\n')
        print(a)
        print('\nB:\n')
        print(b)

        A = self.insSort(a)
        B = self.insSort(b)

        print('\nA (sorted):\n')
        print(A)
        print('\nB (sorted):\n')
        print(B)

        #Combine and insert sort the lists
        X = self.insSort(A + B)
        X = self.xorSort(X)

        print('\nXOR:\n')
        print(X)

    #Sort the list using an insertion sort
    def insSort(self, nums):

        #Compare the positions in the array
        for i in range(1, len(nums)):

            #The value to be compared
            currentvalue = nums[i]

            #Assign the iterator to a new variable to avoid index errors
            position = i
```

```
55
56              #Position must be greater than zero so the index can't be -1
57              while position > 0 and nums[position - 1] > currentvalue:
58
59                  #Assign the value to a new position
60                  nums[position] = nums[position - 1]
61                  position = position - 1
62
63              #Assign the value to a new position
64              nums[position] = currentvalue
65
66          #Return the sorted array
67          return nums
68
69      #The irand function from project 5
70      def irand(self, n, m):
71          b = list(range(n))
72          b = random.sample(range(m), n)
73          return b
74
75      #This method sorts into a single XOR list
76      def xorSort(self, nums):
77
78          #A new array to push to in order to avoid index errors
79          result = nums
80
81          for i in range(1, len(nums) - 1):
82
83              #Iterate over the list to find matches
84              if nums[i - 1] == nums[i]:
85
86                  #Remove both instances of the matching number
87                  result = [j for j in result if j != nums[i]]
88
89          return result
90
91  x = XOR()
```


MINGW32:/C/Users/npaxton/Workspace/Discrete Math
npaxton@CTT02 /C/Users/npaxton/Workspace/Discrete Math
$ python project_8.py
A:
[118, 183, 72, 77, 43, 21, 179, 73, 156, 138, 12, 185, 90, 88, 107, 109, 120, 46
, 22, 124, 119, 59, 9, 50, 164, 166, 60, 134, 122, 25, 57, 106, 196, 116, 26, 19
, 173, 8, 193, 176, 66, 74, 91, 127, 115, 142, 126, 14, 65, 38, 113, 82, 17, 163
, 153, 44, 135, 195, 197, 41, 11, 5, 95, 165, 162, 51, 110, 186, 121, 49, 189, 8
9, 157, 188, 4, 154, 170, 54, 168, 187, 152, 114, 56, 62, 172, 182, 145, 130, 40
, 132, 159, 34, 141, 83, 69, 125, 101, 112, 20, 198]
B:
[66, 176, 190, 143, 26, 137, 146, 163, 55, 130, 157, 14, 44, 100, 128, 108, 23,
122, 20, 168, 117, 24, 158, 172, 171, 147, 119, 131, 111, 75, 4, 34, 125, 9, 10,
192, 76, 71, 90, 83, 17, 33, 113, 141, 162, 63, 54, 67, 153, 123, 36, 160, 148,
106, 30, 154, 31, 81, 85, 166, 51, 49, 193, 37, 32, 112, 152, 177, 155, 40, 3,
101, 0, 86, 174, 181, 99, 79, 195, 185, 151, 94, 139, 180, 178, 8, 129, 6, 150,
97, 80, 82, 95, 7, 188, 196, 179, 169, 52, 65]
A (sorted):
[4, 5, 8, 9, 11, 12, 14, 17, 19, 20, 21, 22, 25, 26, 34, 38, 40, 41, 43, 44, 46,
49, 50, 51, 54, 56, 57, 59, 60, 62, 65, 66, 69, 72, 73, 74, 77, 82, 83, 88, 89,
90, 91, 95, 101, 106, 107, 109, 110, 112, 113, 114, 115, 116, 118, 119, 120, 12
1, 122, 124, 125, 126, 127, 130, 132, 134, 135, 138, 141, 142, 145, 152, 153, 15
4, 156, 157, 159, 162, 163, 164, 165, 166, 168, 170, 172, 173, 176, 179, 182, 18
3, 185, 186, 187, 188, 189, 193, 195, 196, 197, 198]
B (sorted):
[0, 3, 4, 6, 7, 8, 9, 10, 14, 17, 20, 23, 24, 26, 30, 31, 32, 33, 34, 36, 37, 40
, 44, 49, 51, 52, 54, 55, 63, 65, 66, 67, 71, 75, 76, 79, 80, 81, 82, 83, 85, 86
, 90, 94, 95, 97, 99, 100, 101, 106, 108, 111, 112, 113, 117, 119, 122, 123, 125
, 128, 129, 130, 131, 137, 139, 141, 143, 146, 147, 148, 150, 151, 152, 153, 154
, 155, 157, 158, 160, 162, 163, 166, 168, 169, 171, 172, 174, 176, 177, 178, 179
, 180, 181, 185, 188, 190, 192, 193, 195, 196]
XOR:
[0, 3, 5, 6, 7, 10, 11, 12, 19, 21, 22, 23, 24, 25, 30, 31, 32, 33, 36, 37, 38,
41, 43, 46, 50, 52, 55, 56, 57, 59, 60, 62, 63, 67, 69, 71, 72, 73, 74, 75, 76,
77, 79, 80, 81, 85, 86, 88, 89, 91, 94, 97, 99, 100, 107, 108, 109, 110, 111, 11
4, 115, 116, 117, 118, 120, 121, 123, 124, 126, 127, 128, 129, 131, 132, 134, 13
5, 137, 138, 139, 142, 143, 145, 146, 147, 148, 150, 151, 155, 156, 158, 159, 16
0, 164, 165, 169, 170, 171, 173, 174, 177, 178, 180, 181, 182, 183, 186, 187, 18
9, 190, 192, 197, 198]
npaxton@CTT02 /C/Users/npaxton/Workspace/Discrete Math
$