Name: Daniel Throop COSC 3100
MUID: 006165173 **Programming Assignment 3** February 16, 2021

0.1 Explanation of Code

Here we have a bibd solver of design (6, 10, 3, 2, 5). We use the idea of backtracking and the program call stack to solve this problem. First, we initialize our blocks as lists within a list. We then define what constitutes as a complete block, a valid block, and how to find the first empty spot in a block. The main logic of the program guesses numbers in ascending order to speed up the result. Lastly, we test the run time of the definition.

```python
import time

# BIBD conditions
nr_blocks = 10
pts_per_block = 3
nr_elements = 6
distinct = 2
blks_with_point = 5

blocks = [pts_per_block*[0] for _ in range(nr_blocks)]

# should work (not tested)
def is_complete(blocks):
    for block in range(10):
        for i in range(3):
            if blocks[block][i] == 0:
                return False
    return True

# both conditions tested and working
def is_valid(blocks):
  # count data structures
  elCount = [0]*nr_elements
  pairs = {"12": 0, "13": 0, "14": 0, "15": 0, "16": 0, "23": 0, "24": 0, "25": 0, "26": 0, "34": 0, "35": 0, "36": 0, "45": 0, "46": 0, "56": 0}

  # check that an element appears exactly 5 times in different blocks
  for block in range(10):
    for i in range(3):
      # if the given position isn't a zero we want to investigate it
      if blocks[block][i] != 0:
        # record that we have seen the element
        elCount[(blocks[block][i]) - 1] += 1

```

```
# determine if the element count is valid
for num in elCount:
  # if a element appears more than 5 times accross the blocks the solution is invalid
  if num > 5:
    return False

# check that any pair of elements is in two blocks
for block in range(10):
  if blocks[block][1] != 0:
    pair1 = str(blocks[block][0]) + str(blocks[block][1])
    pairs[pair1] += 1
  if blocks[block][2] != 0:
    pair2 = str(blocks[block][1]) + str(blocks[block][2])
    pairs[pair2] += 1
    pair3 = str(blocks[block][0]) + str(blocks[block][2])
    pairs[pair3] += 1
  else:
    # do nothing its a pair with a zero in it
    pass

# determine if the pair count is valid
```

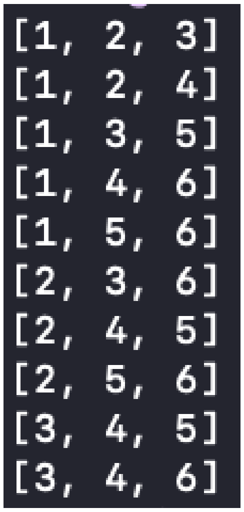
```
  for num in pairs.values():
    # if a given pair appears more then two times the solution is invlaid
    if num > 2:
      return False

  # all validity tests passed
  return True

# appears to be working (somewhat tested)
def find_first_empty(blocks):
  for block in range(10):
    for i in range(3):
      if blocks[block][i] == 0:
        return block, i

# prints blocks accordingly
def print_it(blocks):
  for block in range(10):
    print(blocks[block])

# main logic
```

```
def bibd(blocks):
  if is_complete(blocks):
    return blocks
  # blk is a block , i an index, this is the first block with a 0
  blk, i = find_first_empty(blocks)
  for num in range(max(blocks[blk])+1, nr_elements+1):
    blocks[blk][i] = num
    if is_valid(blocks):
      result = bibd(blocks)
      if is_complete(result):
        return result
    blocks[blk][i] = 0
  return blocks


# TESTING

# def(s) testing
# print(find_first_empty(blocks))
# print(is_complete(blocks))
# print_it(blocks)
# print(is_valid(blocks))
```

```python
# print_it(blocks)
# [6, 0, 0]

# trial runner
start_time = time.perf_counter()
bibd(blocks)
print(time.perf_counter() - start_time)
```

Algorithm 1: **bibd solver Python 3.9**

0.2 Output

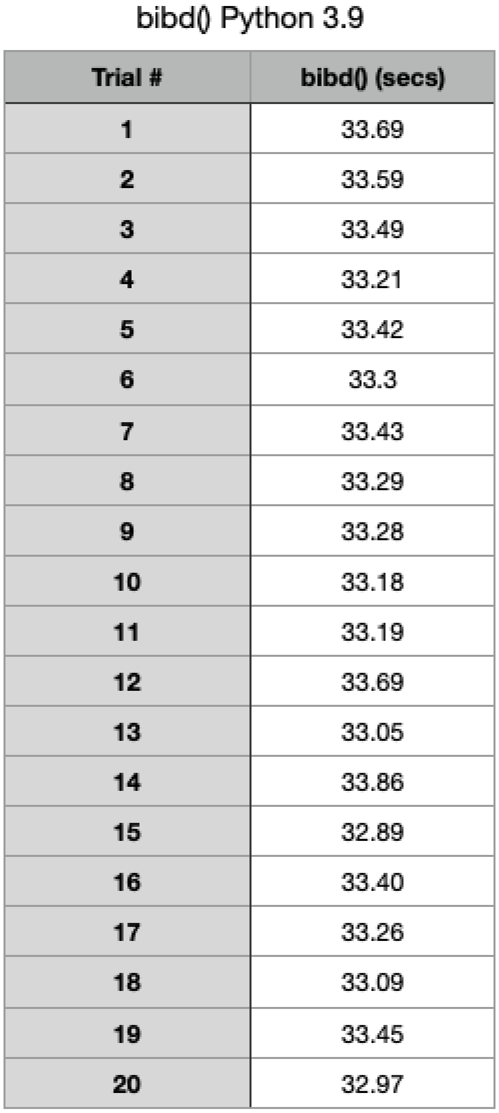


```
[1, 2, 3]
[1, 2, 4]
[1, 3, 5]
[1, 4, 6]
[1, 5, 6]
[2, 3, 6]
[2, 4, 5]
[2, 5, 6]
[3, 4, 5]
[3, 4, 6]
```

Figure 1: Output of Program

0.3 Run time

These run times were captured on a Quad-Core Intel Core i5 at 1.4 GHz.



bibd() Python 3.9

Trial #	bibd() (secs)
1	33.69
2	33.59
3	33.49
4	33.21
5	33.42
6	33.3
7	33.43
8	33.29
9	33.28
10	33.18
11	33.19
12	33.69
13	33.05
14	33.86
15	32.89
16	33.40
17	33.26
18	33.09
19	33.45
20	32.97

Figure 2: Table of run times