HW 10-11

Prog1) This is considerably more complicated than expected. My output represents rows 5-50 and cols 3-49 where each entry is odd -'o' or even '.'

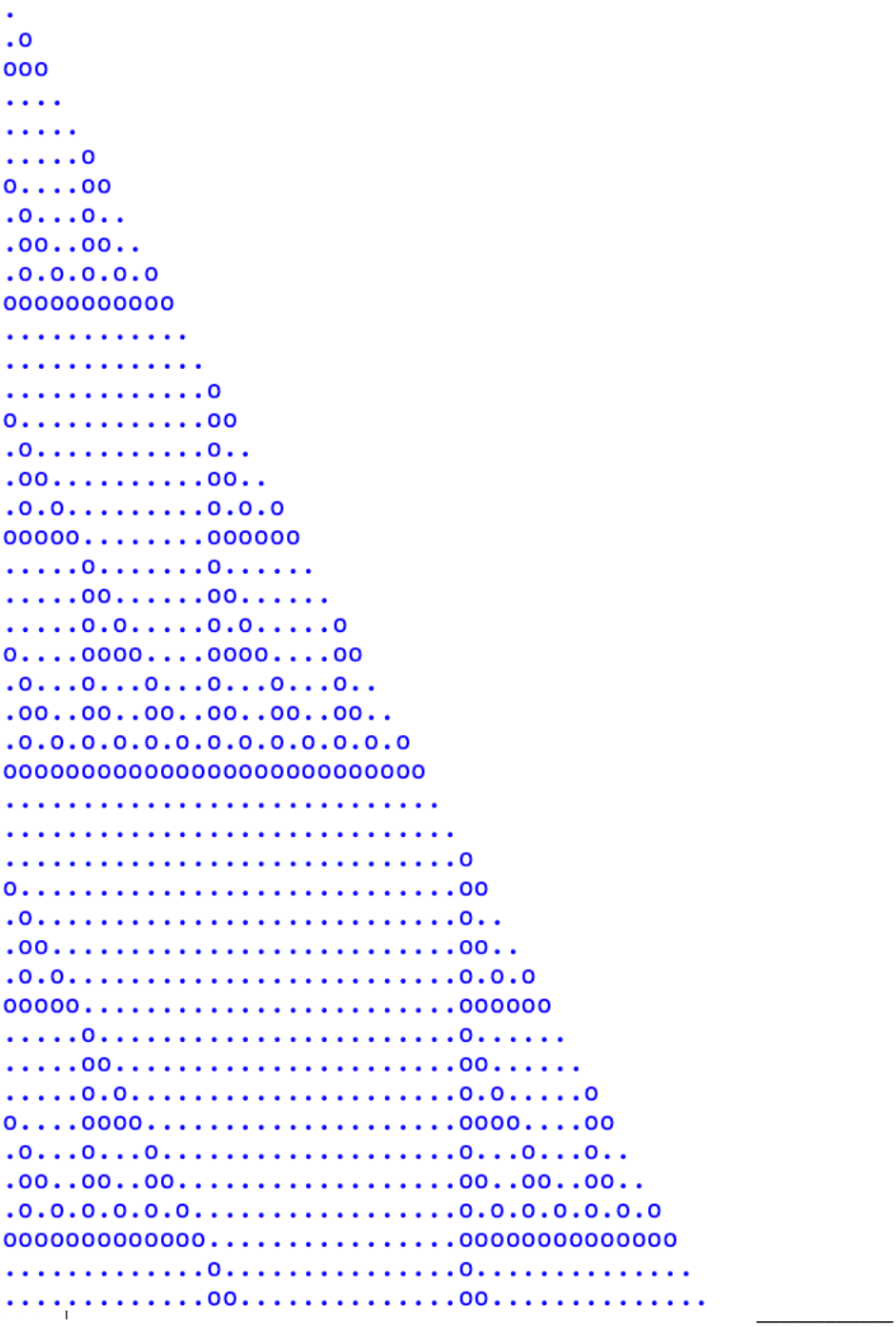
```
# Find where C(n,k) is odd
def fact(n):
    if n==1 or n ==0:
        return 1
    else:
        return n*fact(n-1)

def comb(n,k): #This is the efficient way to calc C(n.k)
    p=1
    k = min(k,n-k)
    for i in range(k):
        p *= n-i
    return p//fact(k)

for i in range(5,50):
    for j in range(3,50):
        x=comb(i,j)

        if x%2==1:
            print('o',end='')
        else:
            print('.',end='')
    print("")
```

See below o's signify odds and rows and cols signify values for n and k. so each row is C(n,k) where n is fixed and k varies from 3 to n-1. Wow-this was an interesting problem for me. Look at the row for n=31- every single value is odd!!!!!!!!!!! Is there a pattern? It turns out to have something to do with the Sierpinski Triangle-look it up



Prog2)

```
# Find empirical prob that 2 random ints <=n have gcd=1
import random
# The Euclidean alg
def gcd(y, x):
    while x>0:
        oldx=x
        x=y%x
        y=oldx
        #print(oldx,x,y)
    return oldx

n=100
count=0
matches=0
for i in range(100000): #I generate 100000 random pairs
    x= random.randint(1,n)
    y= random.randint(1,n)
    if (x==y):
        matches +=1
    if gcd(x,y)==1:
        count+=1


print(count, matches). The answer should show this is close to 6/pi^2
```

Prog3)

```
# Find the next permutation in lexicographic order
def next_perm(list1):
    n = len(list1)-1  # This is the last index ie a(n-1)
    j = n-1
    # Find the first index j that is a decrease-right to left
    while list1[j+1] < list1[j]:
        j -= 1

    #print("j",j)

    # Now find the k that is the smallest index to the right
    # of a(j) with a(j)<a(k)
    k=n #added
    while list1[j]>list1[k]:
        k-=1
    #print("min",min,"k",k)

    # Swap a(j) with a(k)
    list1[k], list1[j] = list1[j], list1[k]

    # Now this will leave the values to the right of firstleft
```

```
    # in descending order. so now just start swapping values

    r = n
    s = j+1
    while r > s:
        list1[s], list1[r] = list1[r], list1[s]
        r -= 1
        s += 1
    return list1


list1 = [1,2,3,4,5]
#print(next_perm(list1))
for i in range(120):
    print(list1)
    list1=next_perm(list1)
    print()
```

Prog 4)

```
# Find the next r combinations of 1,2,...n
A=[1,2,3,4]

# This func Finds the first index
# from the right  you can add to
def find_index(A,n):
    r=len(A)
    i=r

    while (A[i-1]==n-r+i):
        i += -1
        #print(i)
    return i-1


# Increase the value of A[find_index] by 1 and
# each subsequent value by 1 more
def next_combo(A,n):
    r=len(A)
    j=find_index(A,n)

    A[j] += 1
    for k in range(j,r-1):
        A[k+1]=A[k]+1 # Everything to the right is
        #the next number
    return A
```

```
for i in range(14):
    A = next_combo(A,6)
    print(A)

next_combo(A,6)
```

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

Prog5)

```
#Next Binary number -ie adding 1
A=[1,0,1,0,1,1,1,1,1]
def add_1(A):
    i=-1
    while A[i] ==1:
        A[i]=0 # Change values to 0 as you search for 1st 0
        i -=1
    A[i]=1 # Once the loop is done you have found where A[i]==0
    return A
print(A)
print(add_1(A))
```

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

Prog 6

```
# Generate all bit strings 3 bits and 5 0's
# Find the next r combinations of 1,2,...n
A=[1,2,3,4,5,6,7,8]

# This func Finds the first index
# from the right  you can add to
def find_index(A,n):
    r=len(A)
    i=r
    while (A[i-1]==n-r+i):
        i += -1
        #print(i)
    return i-1

# Increase the value of A[find_index] by 1 and
```

```
# each subsequent value by 1 more
def next_combo(A,n):
    r=len(A)
    j=find_index(A,n)

    A[j] += 1
    for k in range(j,r-1):
        A[k+1]=A[k]+1 # Everything to the right is
        #the next number
    return A

#The following function takes a set like A= 2,4,5
# and produces a bit string 01011000
def r_comb_to_bit_str(A):
    bit_str=[0]*8
    for i in range(len(A)):
        bit_str[A[i]-1] = 1
        # A[2]=5 so bit_str[4]=1
    return bit_str
r=[1,2,3]

# Generate all 56 C(8,3) bit strings
print(r_comb_to_bit_str(r))
for i in range(55):
    r=next_combo(r,8)
    print(r_comb_to_bit_str(r))
```

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

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

1) abc bca cba cab acb bac

2) 7!

3) 6!

4) - ——— b) 123 234 345 a) $5 \cdot 4 \cdot 3$
134 235
135 245
145
125
124

8) 5! 9) $12 \cdot 11 \cdot 10$

11) a) $C(10,4)$

b) $C(10,4) + C(10,3) + C(10,2) + C(10,1) + C(10,0)$

c) $2^{10} - (C(10,3) + C(10,2) + C(10,1) + C(10,0))$

d) $\binom{10}{5}$

---

13) [m] [w] ÷ — [w]

$2(n!)^2$

27) a) $\binom{25}{4}$ b) $25 \cdot 24 \cdot 23 \cdot 22$

35) 01 ← 8 pairs like this with 2 xtra 1's
so there are 10 spots to place the 2 1's
ie $C(10,2)$

36) 011

we have 5 of these with 4 1's left over

we thus have 9 spots to place 011's or single 1's

Thus $\boxed{C(9,4)}$

37) $C(10,3)+C(10,4)+C(10,5)+C(10,6)+C(10,7)$

2) 156423 165432 231456 231465 -- etc

6) a) 1324 b) 51234 c) 13254

d) 612354 e) 1623574 f)