1.

If each cubicle contains less than [n/m] programmers
Then our total number of programmers is less than m*(n/m)
= n
So there must be at least one cubicle with at least n/m programmers

2.

Since queue's are FIFO (First in First Out) data structures running at worst case time would involve a pointer corresponding to the last item (b) in the list. Since the linked list is circular but not doubly linked getting from the tail (enqueue) to the head (dequeue) would involve traversing every item in the list +1 to move back to the tail to enqueue. O(n+1)?

3.

Proof:
Assume the number of primes is finite

Let P be the largest prime. Consider the number P!+1

Let Q = P!+1

It can be observed that Q mod 2 = 1, Q mod 3 = 1, Q mod 7 = 1 ......

no prime is a factor of Q

Thus Q is a prime greater than P so primes cant be finite.

4.

Use Birthday Paradox
Find the chance that it wont be distinct and minus it from the probability it will be

We can only generate numbers up to n
we have already generated (i - 1) values

Let P be the probability of a distinct value.
P(i) = 1 - (i - 1)/n

5. (a) O(n^2)

(b) O(n(logn))

(c) Initialise array with numbers 1 - n | O(n)

Generate Random number

```
Swap A{i} with A{r} | O(n)
```

6.

Algorithmn:

```
n = 1


while (room not found)
{
     step left n

        if not room
          n = n*2
        else room found

     step right n

       if not room
         n = n*2
       else room found
}
```

By doubling our steps in the opposite direction if our room isnt found
i.e 1(2)+2(2)+4(2)+8(2)+16(2)...
we cover a significant amount of distance without wasting time and energy
backtracking an insignificant number of steps.

Using a fairly ugly visual representation it can be seen that the door
will exist between the functions f(x) and f(x+2)

```
|_____(2x^-1)______|___(2x^-2)__n__(2x^i)___|
                   0
```

or 2x^i-2 < n <= 2x^i