a) Some of the test cases for the question are:

1. For x=10:
Explanation- The value of the answer would be 5 as:
1! = 1 (not divisible by 10)
2! = 2 (not divisible by 10)
3! = 6 (not divisible by 10)
4! = 24 (not divisible by 10)
5! = 120
since, 120/10= 12. Hence, it is divisible by 10. So, i=5 for x=10.

2. For x=9:
Explanation- The value of the answer would be 6 as:
1! = 1 (not divisible by 9)
2! = 2 (not divisible by 9)
3! = 6 (not divisible by 9)
4! = 24 (not divisible by 9)
5! = 120 (not divisible by 9)
6! = 720
since, 720/9= 80. Hence, it is divisible by 9. So, i=6 for x=9.

3. For x=8:
Explanation- The value of the answer would be 4 as:
1! = 1 (not divisible by 8)
2! = 2 (not divisible by 8)
3! = 6 (not divisible by 8)
4! = 24
since, 24/8= 3. Hence, it is divisible by 8. So, i=4 for x=8.

4. For x=12:
Explanation- The value of the answer would be 4 as:
1! = 1 (not divisible by 12)
2! = 2 (not divisible by 12)
3! = 6 (not divisible by 12)
4! = 24
since, 24/12= 2. Hence, it is divisible by 12. So, i=4 for x=12.

5. For x=15:
Explanation- The value of the answer would be 5 as:
1! = 1 (not divisible by 12)
2! = 2 (not divisible by 12)
3! = 6 (not divisible by 12)
4! = 24 (not divisible by 12)
5! =120
since, 120/15= 8. Hence, it is divisible by 15. So, i=5 for x=15.

b) What value of x, results in the factorial calculation to overflow?

Explanation- The max capacity of a 64 bit register is 2^64 -1 = 1.84 * 10^19
And the largest factorial just under this is 20! = 2.43 * 10^18

Considering, 21, we won’t have to use this as 21 can be satisfied by 7!
Similarly, 22 can be satisfied by 11! But as 23 is a prime number, it can only be satisfied by 23! which will cause overflow. Hence, for 64-bit: the x that will cause overflow will be equal to 23.
Similarly, for:
32-bit, x=13
16-bit, x=11
8-bit, x=7.