Question 1

How big do you need to make d so that this command shows that x = 0.1 is not represented exactly? (Experiment with different values of d and find the smallest.)

```
x = .1
```

```
x =
     1.000000000000000e-01
```

```
for d = 1:55
    fprintf('%.*e\n', d, x);
end
```

```
1.0e-01
1.00e-01
1.000e-01
1.0000e-01
1.00000e-01
1.000000e-01
1.0000000e-01
1.00000000e-01
1.000000000e-01
1.0000000000e-01
1.00000000000e-01
1.000000000000e-01
1.0000000000000e-01
1.00000000000000e-01
1.000000000000000e-01
1.0000000000000001e-01
1.0000000000000006e-01
1.0000000000000056e-01
1.0000000000000555e-01
1.0000000000000005551e-01
1.0000000000000055511e-01
1.0000000000000555112e-01
1.0000000000000005551115e-01
1.0000000000000055511151e-01
1.0000000000000555111512e-01
1.0000000000000005551115123e-01
1.0000000000000055511151231e-01
1.0000000000000555111512313e-01
1.0000000000000055511151231126e-01
1.0000000000000555111512312578e-01
1.0000000000000055511151231257827e-01
1.0000000000000555111512312578270e-01
1.0000000000000055511151231257827021e-01
1.0000000000000555111512312578270212e-01
1.0000000000000055511151231257827021182e-01
1.0000000000000555111512312578270211816e-01
1.0000000000000055511151231257827021181583e-01
1.0000000000000555111512312578270211815834e-01
1.0000000000000055511151231257827021181583405e-01
1.0000000000000555111512312578270211815834045e-01
```

```
1.000000000000000055511151231257827021181583404454e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
1.000000000000000055511151231257827021181583404541e-01
```

The answer is 54.

Question 2

Write a function that converts a natural number (nonnegative integer) into an array of its decimal digits.

```
uint_to_digits(12345)
```

```
ans = 1×5
     1     2     3     4     5
```

```
uint_to_digits(2^53)
```

```
ans = 1×16
     9     0     0     7     1     9     9     2     5     4     7     4     0 ···
```

```
uint_to_digits(2^53 + 1)
```

```
ans = 1×16
     9     0     0     7     1     9     9     2     5     4     7     4     0 ···
```

```
uint_to_digits(2^53 + 2)
```

```
ans = 1×16
     9     0     0     7     1     9     9     2     5     4     7     4     0 ···
```

The reason why the first two are equal is because of how computers store float numbers. The gap between the float representation for 2^53 and the next number in the natural number line is greater than 1, so any value 1 and lower is rounded to 2^53 and any value 1.00...01 and higher is rounded to 2^53 + 2

Question 3

Write a function that converts an array of decimal digits into a natural number.

```
format long e;
digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 2])
```

```
ans =
     9.007199254740992e+15
```

```
digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 3])
```

```
ans =
     9.007199254740992e+15
```

```
digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 4])
```

```
ans =
     9.007199254740994e+15
```

Same reason as question 2. The number becomes so large that there is a skip in the IEEE756 double precision float representation at this number 9007199254740992, which is equivalent to 2^53.

Question 4

Write a function that implements addition of natural numbers.

```
addn([2 3 4 6], [9 9 9 9])
```

```
ans = 1×5
     1     2     3     4     5
```

Question 5

Write a function that implements multiplication of natural numbers.

```
multn([1 5], [8 2 3])
```

```
ans = 1×5
     1     2     3     4     5
```

```
function X = uint_to_digits(x)
    X = zeros(1, floor(log10(x)) + 1);
    for i = floor(log10(x)):-1:0
        X(1, i + 1) = mod(x, 10);
        x = floor(x/10);
    end
end

function x = digits_to_uint(X)
    x = 0;
    for i = length(X):-1:1
        x = x + (X(i) * 10.^(length(X) - i));
    end
end

function s = addn(a, b)
    s = zeros(1, max(length(a),length(b)) + 1);
    carry = 0;
    for i = length(s)-1:-1:1
```

```
        sum = a(i) + b(i) + carry;
        remainder = mod(sum, 10);
        s(i + 1) = remainder;
        carry = floor(sum/10);
    end
    s(1) = carry;
end

function s = multn(a, b)
    s = [];
    for i = length(b):-1:1
        carry = 0;
        j = 0;
        local = zeros(1, length(b) - i);
        for j = length(a):-1:1
            num = b(i) * a(j) + carry;
            local = [mod(num,10), local];
            carry = floor(num/10);
        end
        s = [zeros(1, max(0,length(local) - length(s))), s];
        s = addn(s, local);
    end
    lastCarry = [carry, zeros(1, length(b) + 1)];
    s = addn(s, lastCarry);
    if s(1) == 0
        s = s(2:length(s));
    end
end
```