# CSCU9V4 Practical 2
# To `scanf` or not to `scanf`

## Introduction

In any programming language there are many ways to solve a problem. In C there are often many more than in other languages. The `scanf` function is subtle, incredibly powerful, and hence a little dangerous. Let's turn these features to our advantage.

## Background

A classic `first program' is to separate the digits of a number. A long solution might look like this,

```
#include<stdio.h>
int main()
{
  int num,temp,factor=1;

  printf("Enter a number: ");
  scanf("%d",&num);

  temp=num;
  while(temp){
      temp=temp/10;
      factor = factor*10;
  }

  printf("Each digits of given number are: ");
  while(factor>1){
      factor = factor/10;
      printf("%d ",num/factor);
      num = num % factor;
  }

  return 0;
}
```

While correct, many would argue it is too complex and too much work. A simpler solution then,

```
int num, digit;
…
while (num > 0) {
 digit = num % 10;
 printf("%d ",num);
 num /= 10;
}
```

Sample output from both solutions:

```
Enter a number: 123
Each digits of given number are: 1 2 3
```

## Turning Things Around

Say instead the number of digits is known in advance, and that the goal is to reverse the digit order, eg. 123 → 321. A two-digit solution might look like,

```
int n;

printf("Enter a two-digit number: ");
scanf("%d", &n);

printf("The reversal is: %d%d\n", n % 10, n / 10);
```

This is what we in CS call a `naïve' approach, ie. an intuitive and natural first-pass solution. What if there are more than two digits? The naïve approach is valid, and worth trying. Though even at three digits the naïve approach quickly grows unmanageable with arithmetic (try it!).

**Your goal:** Design 3-digit solution without using any arithmetic. (Hint: use `scanf`.)

## Pushing the Limits

Some would say that reversing digits is a rather artificial example. A more useful example for those wishing a bit more of a challenge, then: **Retrieve and check the validity in any UPC code**.

All UPCs consist of an 11-digit sequence followed by a check digit. Consider the example UPC of "036000241457". The last digit is the check digit "7", and if the other numbers are correct then the check digit calculation must produce 7. The algorithm, by example, is

1. Add the odd number digits: 0+6+0+2+1+5 = 14
2. Multiply the result by 3: 14 × 3 = 42
3. Add the even number digits: 3+0+0+4+4 = 11
4. Add the two results together: 42 + 11 = 53
5. To calculate the check digit, take the remainder of (53 / 10), which is also known as (53 modulo 10), and subtract from 10. Therefore, the check digit value is 7.

Go code!