CSE 3100 Systems Programming	Spring 2024
Homework #1	

Please submit your work to gradescope.

Exercise 1. (50 points) Approximating π
In this exercise, we use the following fact to approximate π.

$$\pi = \sum_{i=0}^{\infty} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right) \frac{1}{16^i}.$$

To approximate π, we modify the above summation so that i is summed from 0 to n, where n is a non-negative integer.

$$\pi \approx \sum_{i=0}^{n} \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right) \frac{1}{16^i}.$$

In our code, we first read n from the standard input as shown below. Add more code below TODO to finish the code.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n, i;

    printf("n = ");
    scanf("%d", &n);

    //TODO
    //add code below



    printf("PI = %.10f\n", pi);
    return 0;
}
```

Note we are **not allowed** to use power functions in this assignment. Think how we can avoid using power functions in the above summation.

Below are outputs from a few example runs. These can be used to check correctness of our code.

```
$ ./pi
n = 3
PI = 3.1415924576
$ ./pi
n = 5
PI = 3.1415926532
```

Exercise 2. (50 points) Happy Numbers
A *happy number* is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number either equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers.

For example, 13 is a happy number since
$1^2 + 3^2 = 1 + 9 = 10$,
$1^2 + 0^2 = 1 + 0 = 1$.

And 85 is an unhappy number since
$8^2 + 5^2 = 64 + 25 = 89$,
$8^2 + 9^2 = 64 + 81 = 145$,
$1^2 + 4^2 + 5^2 = 1 + 16 + 25 = 42$,
$4^2 + 2^2 = 16 + 4 = 20$,
$2^2 + 0^2 = 4 + 0 = 4$,
$4^2 = 16$,
$1^2 + 6^2 = 1 + 36 = 37$,
$3^2 + 7^2 = 9 + 49 = 58$,
$5^2 + 8^2 = 25 + 64 = 89$, which is a repeated result.

It can be shown that if a number is an unhappy number, it will reach 4 in the above process. This can be used to check whether a number is an unhappy number.

Our job is to write code to check whether a positive integer is a happy number. Part of the code is already given. As shown below, we only need to fill in the code under TODO. Note to print out the intermediate numbers to match the expected output.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n;

    printf("n = ");
    scanf("%d", &n);

    int m = n;

    //TODO
    //add code below









    if(n==1) printf("%d is a happy number.\n", m);
    else printf("%d is a NOT a happy number.\n", m);
    return 0;
}
```

Below are a few example runs.

```
$ ./happy
n = 10
1
10 is a happy number.
$ ./happy
n = 101
2
4
101 is a NOT a happy number.
$ ./happy
n = 111
3
9
81
65
61
37
58
89
145
42
20
4
111 is NOT a happy number.
$ ./happy
n = 888
192
86
100
1
888 is a happy number.
```