HW 2

ENGR 21, Fall 2026.

Due Date	Thu, Sep 17
Turn in link	Gradescope code and PDF
URL	`emadmasroor.github.io/E21-F26/Homework/HW2`

What to turn in

For this assignment, you will turn in a combined PDF file as well as multiple code files.

PDF	Code
1	3
2	4.5, 4.9, 4.12, 4.15, 4.16
4.4, 4.7, 4.11	

> **Warning**
>
> For full credit, show how you arrived at your answers; don't just write the correct answer!

1 Converting between number systems

1.1 from Decimal to Binary

1. `100`
2. `1029`
3. `476`
4. 1.4×10^3

1.2 from Binary to Decimal

1. `0b1011011`
2. `0b110010101`
3. `0b1000000001`
4. `0b10111`

1.3 from Decimal to Hexadecimal

1. `1000`
2. `176`
3. `42678`
4. `81`

1.4 from Hexadecimal to Decimal

1. `0x14E3`
2. `0xA10B`
3. `0x1000`
4. `0x1010`

2 Base systems

2.1 A new base: 7

1. Write down all numbers from 1 to 16 in base 7.
2. In base 7, what is the meaning of the symbol `66` ? Explain.
3. What is the largest possible four-'digit' number in base 7? Write the number in words, in decimal form, and in base-7 form.

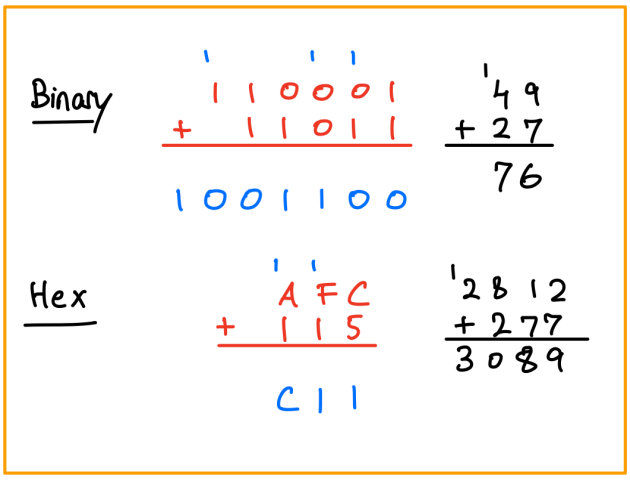
2.2 Addition of binary and hexadecimal numbers

In grade school, you learned how to add two multi-digit numbers by hand. In case, you've forgotten, here's a video explaining this to elementary school students.

Your task in this problem is to carry out multi-'digit' addition, by hand, to binary and hexadecimal numbers. In the process, you will see the concept of 'carrying over' a digit (perhaps you learned the term 'regrouping') applies to numbers other than base ten.

As an example, the following sums have been computed for you below.

- `0b110001` + `0b11011`
- `0xAFC` + `0x115`



2.3 Hex

1. `0xE5A` + `0x85A`
2. `0xEE` + `0x99`
3. `0x100` + `0x20`
4. `0xB0` + `0xB`

2.4 Binary

1. `0b1011011` + `0b11011001`
2. `0b10011` + `0b100110001`
3. `0b10000000000` + `0b1011111`
4. `0b11010001001` + `0b110110110`

3 Conditionals and loops

3.1 Counter variables

Consider the following program, which counts the number of times pin A7 has been tapped and assigns `color1` to pixel number `k`, where `k` is the number of times pin A7 has been tapped. The code crashes if pin A7 is touched more than 10 times.

```
from adafruit_circuitplayground.express import cpx
import time
off = (0,0,0)
color1 = (10,30,10)
color2 = (30,10,10)
cpx.pixels.fill(off)
k = -1
while True:
    time.sleep(0.2)
    if cpx.touch_A7:
        k += 1
        cpx.pixels[k] = color1
        cpx.pixels[k-1] = off
```

Modify this code so that it has the same behavior for up to 10 taps, but for taps 11 through 20, a different color is assigned to the `p`th pixel, where `p` is `k-10`. Your code will still fail after the 21st tap is detected; that's fine.

3.2 The `continue` keyword

The following progam was written with the intention of printing all numbers except for 9. However, the program currently fails at doing this. What one change will ensure that printing '9' is skipped? Turn in the new code.

```
import time
k = 0
while True:
    time.sleep(0.5)
    k += 1
    print(k)
    if k == 9:
        continue
```

3.3 The `break` keyword

In class, we saw that the following code allows us to terminate the inner `while` loop by touching pin A2.

```
from adafruit_circuitplayground.express import cpx
import time
k = 0
while True:
    time.sleep(1)
    k += 1
    print(f"Outer iteration {k}")
    if k % 3 == 0: # k ÷ 3 has remainder 0
        m = 0
        while True:
            time.sleep(1)
            m += 1
            print(f"Outer iteration {k}, inner iteration {m} ")
            if cpx.touch_A2:
                print("terminating")
                break
```

Change the code above so that it meets the following requirements:

- Touching A2 while in the inner loop should terminate inner loop.
- Touching A2 while in the outer loop should not do anything.
- Touching A7 alone while in the inner loop should now **also** terminate inner loop.
- Touching A7 *while pressing button B* while in the outer loop should terminate the outer loop.

3.4 The `time` functions

The function `time.monotonic()` can be used to measure intervals of time. For example, the following code measures how long after the red LED comes on the user presses button B.

```
from adafruit_circuitplayground.express import cpx
import time
cpx.red_led = True
start_time = time.monotonic()
while True:
    time.sleep(0.01)
    if cpx.button_b:
        end_time = time.monotonic()
        interval = end_time - start_time
        break
print(f"Interval was {interval} seconds")
```

Rewrite the code above so that it executes the same functionality without using `break` or `if`.

4 The Farmer Was Replaced

For this part of the assignment, you will have to make progress on the game The Farmer Was Replaced. You will be provided a gift card for the game soon.

Start a new game and complete the following tasks in order. Only the marked tasks require a submission; those that are not in bold are not associated with any submission.

> **Note**
>
> It is important that you avoid unlocking any options other than the ones explicitly listed. If you accidentally unlock more features, you should re-start the game.

1. Harvest (grass) five times
2. Unlock `while`
3. Use the `while` loop to harvest 200 units of hay.
4. Describe, in words, what the drone does using each of these snippets of code.

   - Option 1

   ```
   while True:
      do_a_flip()
      while True:
        harvest()
   ```

   - Option 2

   ```
   while True:
      harvest()
      while True:
        do_a_flip()
   ```

   - Option 3

   ```
   while True:
      harvest()
      do_a_flip()
   ```

   - Option 4

   ```
   while True:
      do_a_flip()
      harvest()
   ```

   - Option 5

   ```
   while True:
      do_a_flip()
   harvest()
   ```

- Option 6

```
while True:
   harvest()
do_a_flip()
```

5. Unlock speed (1/5 only)
6. Write code that executes a flip if a harvest isn't available, and harvests if possible.
7. Explain why, after unlocking speed, the following code no longer works.

```
while True:
   harvest()
```

8. Unlock bush planting
9. Write a program that causes the drone to keep doing flips as it waits for bushes to grow, and harvests them when they're ready to be harvested. It immediately plants a bush once it is done harvesting.
10. Expand the farm. It should now be size 3 x 1.
11. Describe in words what the following code does.

```
while True:
    if can_harvest():
      harvest()
      plant(Entities.Bush)
    else:
      move(North)
```

12. We would like to write a program that makes good use of our agricultural resources by planting bushes on two of the plots and grass on the middle plot. It may be helpful to use the following snippet of code for this purpose.

```
clear()
plant(Entities.Bush)
move(North)
move(North)
plant(Entities.Bush)
move(North)
```

    Taking inspiration from the above code if necessary, write a script that, when run, continuously plants bushes on the two outer plots, allows grass to grow on the middle spot, and harvests all three plots one by one for ever. The first line in your code should be `clear()`

13. Unlock carrots.
14. Expand the farm one more time, getting to a 3 x 3 size.
15. Without using `for` loops, achieve the functionality shown by the following gif. It is possible to do so using just 9 lines of code. The first line should be `clear()`.

16. The following code, when executed, produces the functionality shown in the gif below.

```
clear()
till()
move(East)
till()
move(East)
till()
move(East)
while True:
  # First row
  if can_harvest():
    harvest()
  plant(Entities.Carrot)
  move(East)
  if can_harvest():
    harvest()
  plant(Entities.Carrot)
  move(East)
  if can_harvest():
    harvest()
  plant(Entities.Carrot)
  move(East)
  move(North)
  # Second row: Grass
  if can_harvest():
    harvest()
  move(East)
  if can_harvest():
    harvest()
  move(East)
  if can_harvest():
    harvest()
  move(East)
  move(North)
  # Third row
  if can_harvest():
    harvest()
  plant(Entities.Bush)
  move(East)
  if can_harvest():
    harvest()
  plant(Entities.Bush)
  move(East)
  if can_harvest():
    harvest()
  plant(Entities.Bush)
  move(East)
```

```
move(North)
```

Modify the above code so that carrots are in the middle row and grass in the bottom row. The bushes should still be in the top row.