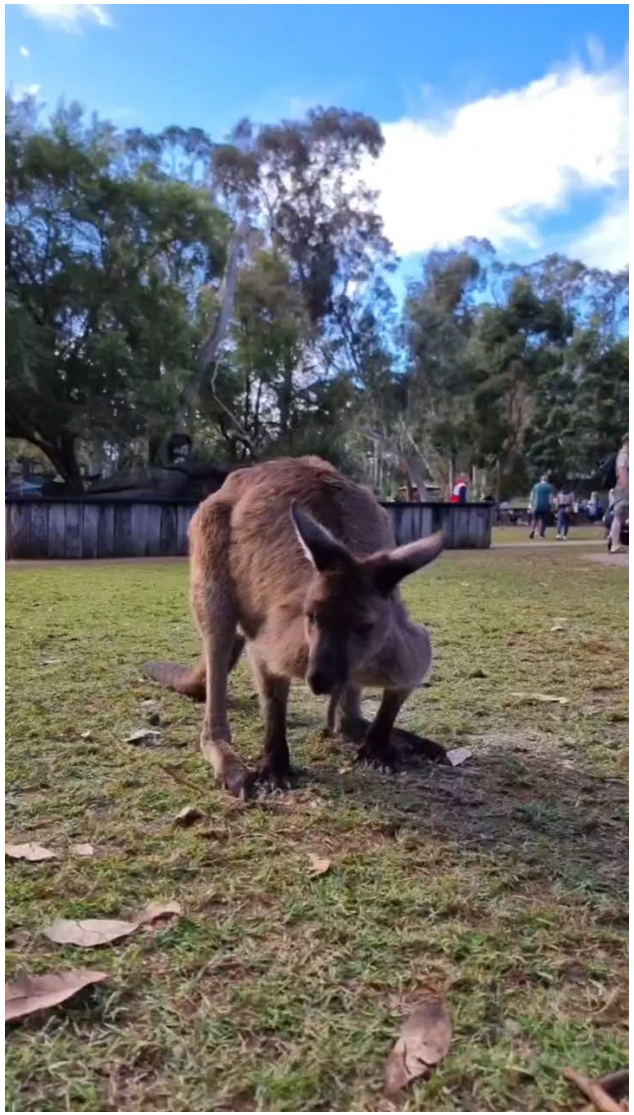
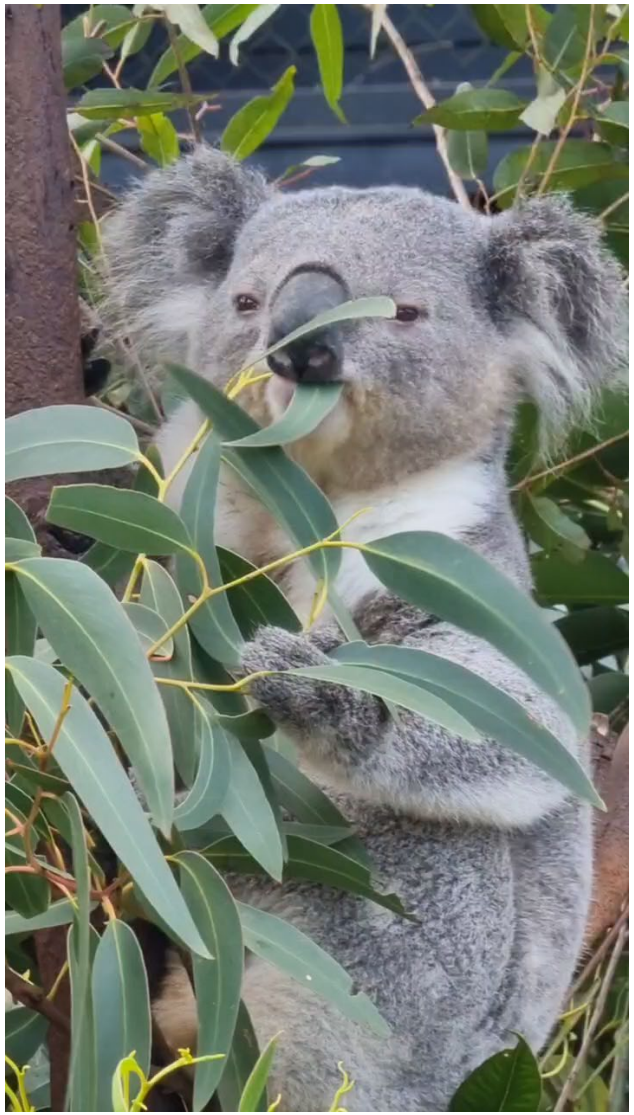
videos I took recently





COMP1911 Week 3!

T11X: 11am – 2pm

Tutors: William (me!) + Daniel

My GitHub:

https://github.com/william-o-s/unsw_comp1511_tutoring

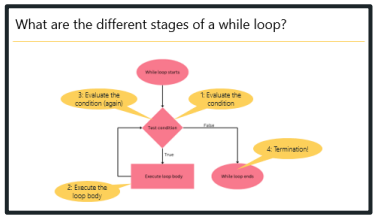
Tutorial Agenda:

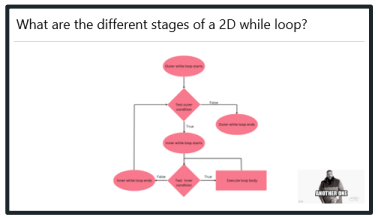
Part 1

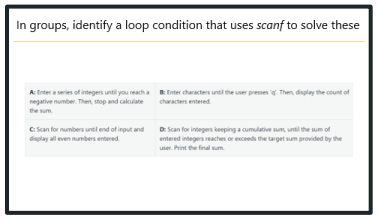
Part 2

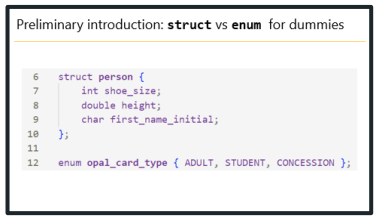
Part 3

Part 4

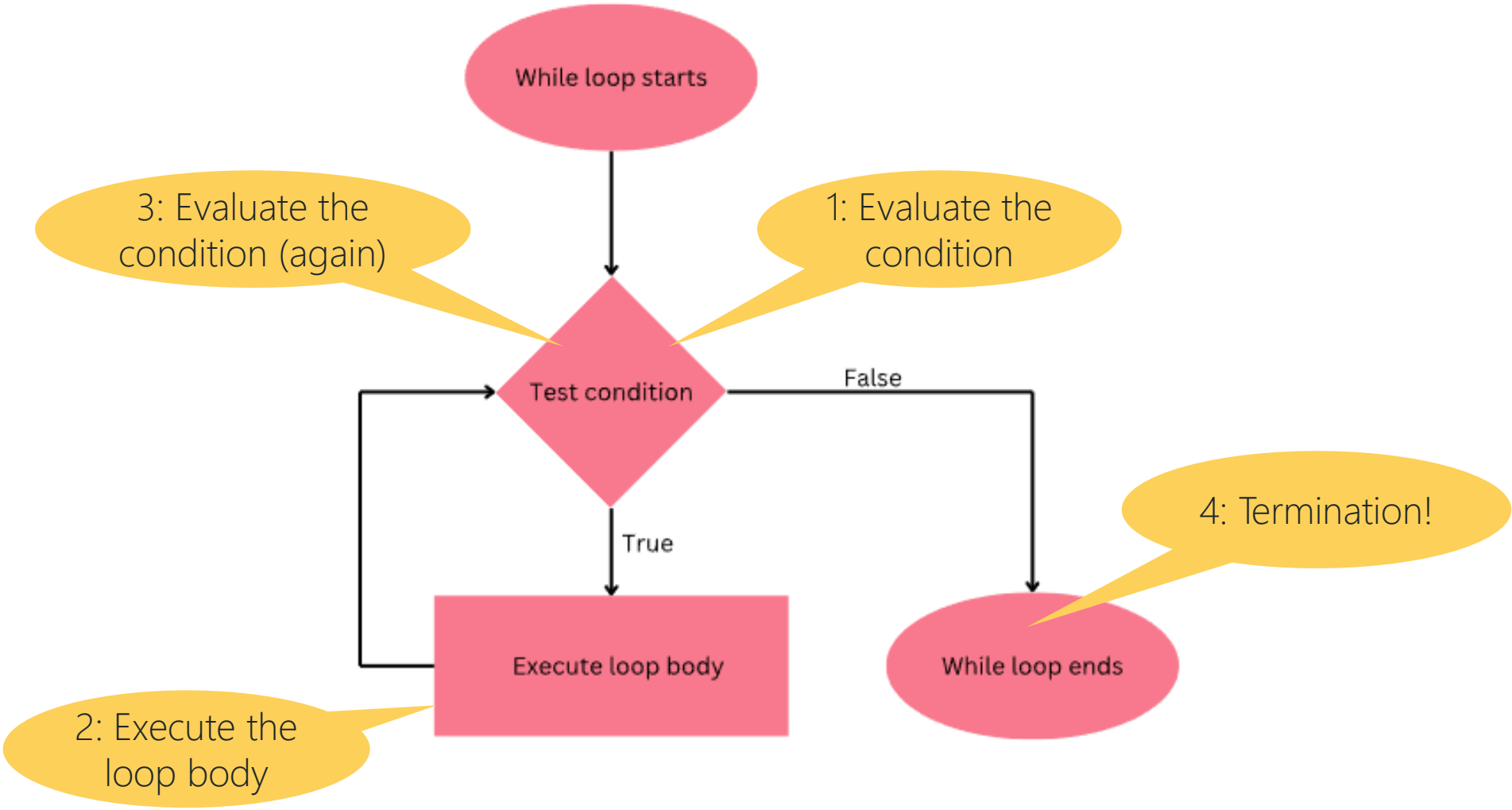




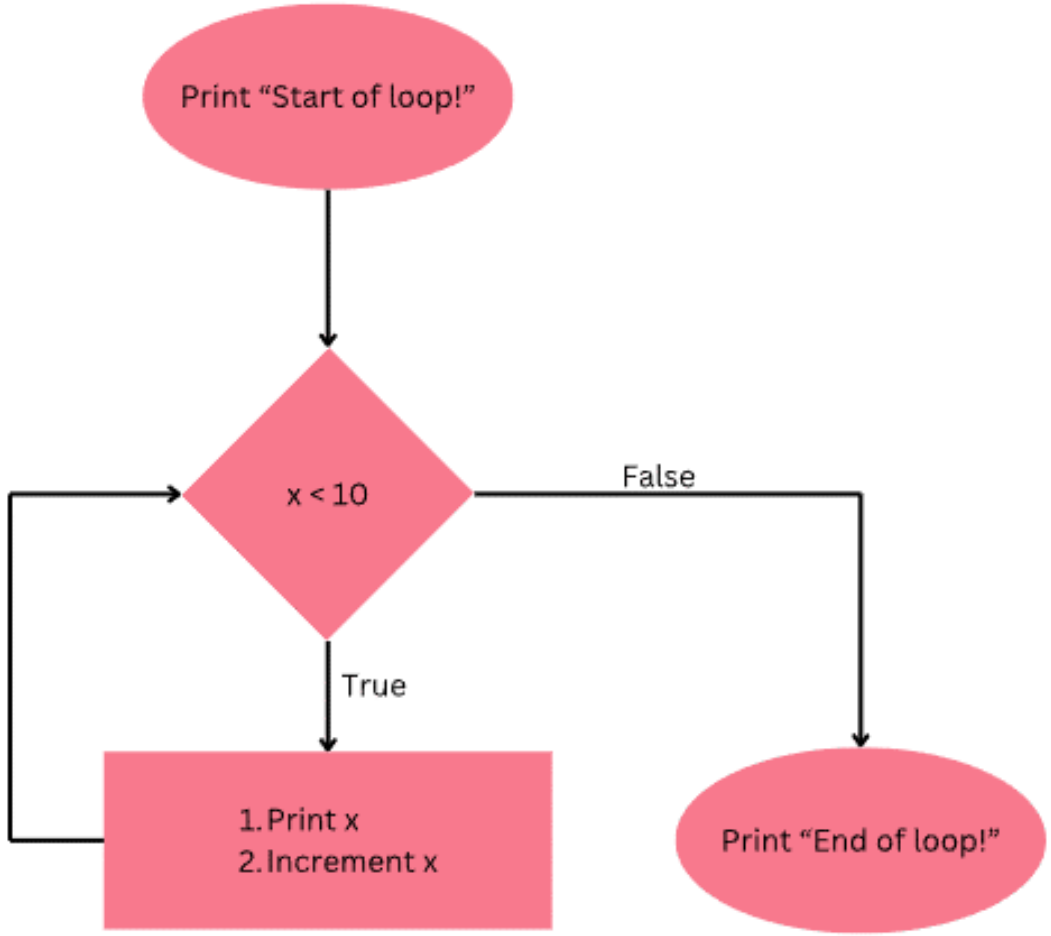




What are the different stages of a while loop?



Let’s convert this flowchart into code!



In groups, guess and verify the output of these snippets

A

```c
#include <stdio.h>

int main(void) {
    int i = 0;
    while (i < 32) {
        printf("%d\n", i);
        i = i + 2;
    }
    return 0;
}
```

B

```c
#include <stdio.h>

int main(void) {
    int i = 5;
    while (i >= 0) {
        printf("%d\n", i);
        i--;
    }
    return 0;
}
```

C

```c
#include <stdio.h>

int main(void) {
    int i = 0;
    int keep_going = 1;
    while (keep_going == 1) {
        if (i > 3) {
            keep_going = 0;
        }
        i++;
    }
    printf("%d\n", i);
    return 0;
}
```

D

```c
#include <stdio.h>

int main(void) {
    int i;
    while (i > 0) {
        printf("%d\n", i);
        i--;
    }
    return 0;
}
```

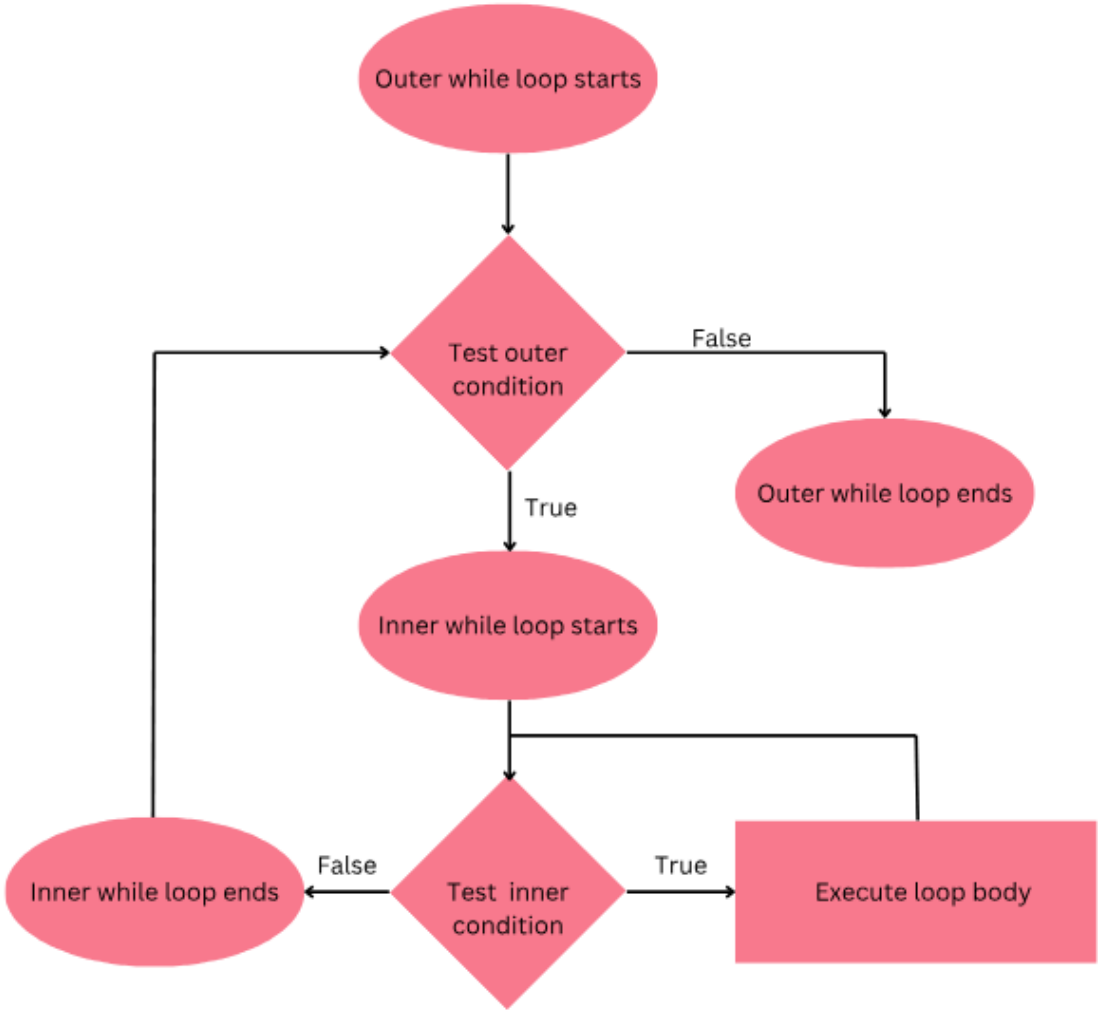
E

```c
#include <stdio.h>

int main(void) {
    int i = 0;
    int max = 32;
    while (i < max) {
        printf("%d\n", i);
        max = max + 2;
    }
    return 0;
}
```

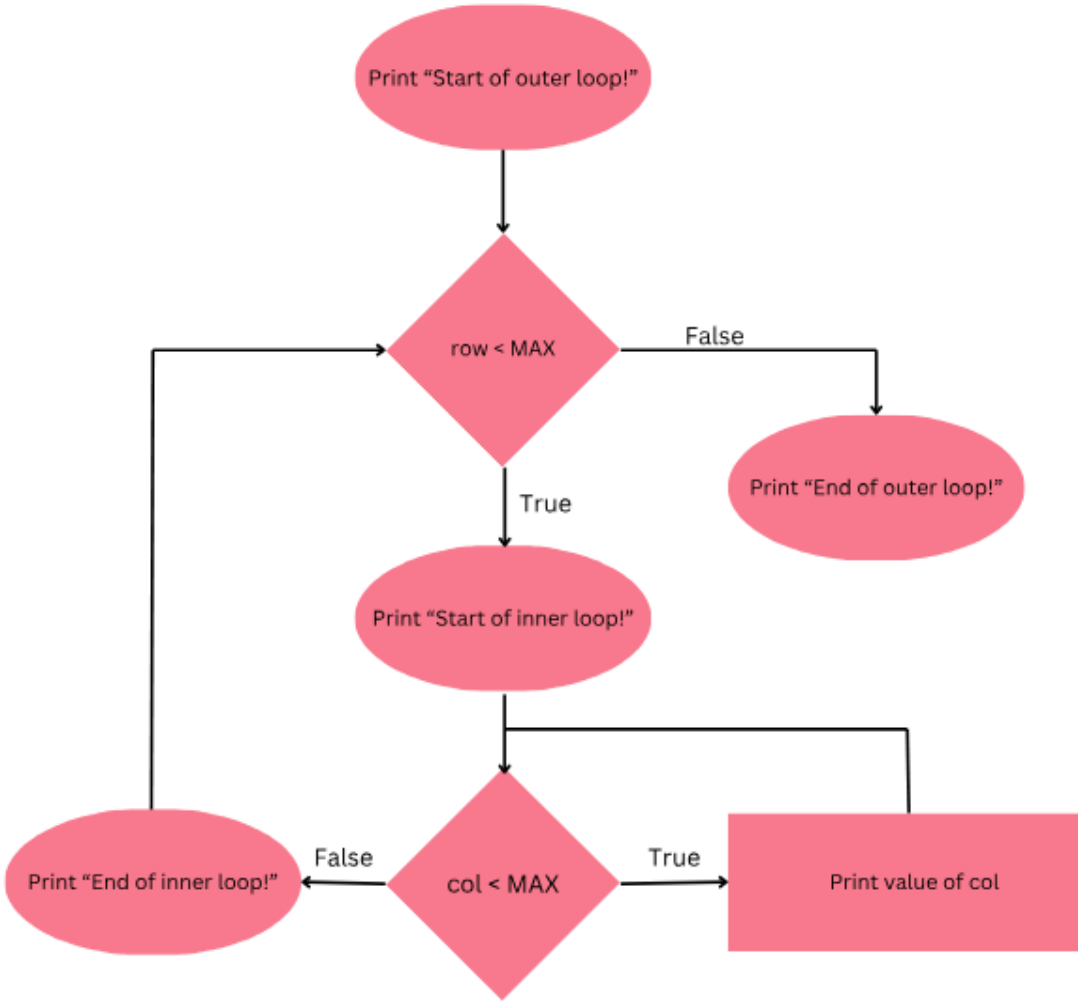
F

```c
#include <stdio.h>

int main(void) {
    int i = 0;
    int keep_going = 0;
    while (keep_going == 1) {
        if (i > 3) {
            keep_going = 0;
        }
        i++;
    }
    printf("%d\n", i);
    return 0;
}
```

What are the different stages of a 2D while loop?



Let’s convert this flowchart into code!



Let’s go backwards – match these outputs to snippets

A

```
0XXX
X0XX
XX0X
XXX0
```

B

```
0X0X
0X0X
0X0X
0X0X
```

C

```
0X00
XXXX
0X00
0X00
```

D

```
XXXX
X00X
X00X
XXXX
```

1

```c
#include <stdio.h>

#define SIZE 4

int main(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (col != 1 && row != 1) {
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
    return 0;
}
```

2

```c
#include <stdio.h>

#define SIZE 4

int main(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (row == col) {
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
    return 0;
}
```

3

```c
#include <stdio.h>

#define SIZE 4

int main(void) {
    int row = 0;
    while (row < SIZE) {
        printf("X");
        int col = 1;
        while (col < 3) {
            if (row == 0 || row == 3) {
                printf("X");
            } else {
                printf("O");
            }
            col++;
        }
        printf("X");
        row++;
        printf("\n");
    }
    return 0;
}
```

4

```c
#include <stdio.h>

#define SIZE 4

int main(void) {
    int row = 0;
    while (row < SIZE) {
        int col = 0;
        while (col < SIZE) {
            if (col % 2 == 0) {
                printf("O");
            } else {
                printf("X");

            }
            col++;
        }
        row++;
        printf("\n");
    }
    return 0;
}
```

In groups, identify a loop condition that uses *scanf* to solve these

A: Enter a series of integers until you reach a negative number. Then, stop and calculate the sum.	**B:** Enter characters until the user presses 'q'. Then, display the count of characters entered.
C: Scan for numbers until end of input and display all even numbers entered.	**D:** Scan for integers keeping a cumulative sum, until the sum of entered integers reaches or exceeds the target sum provided by the user. Print the final sum.

Preliminary introduction: **`struct`** vs **`enum`** for dummies

```
 6    struct person {
 7        int shoe_size;
 8        double height;
 9        char first_name_initial;
10    };
11
12    enum opal_card_type { ADULT, STUDENT, CONCESSION };
```

What differences can you see?

```
 6   struct person {
 7       int shoe_size;
 8       double height;
 9       char first_name_initial;
10   };
11
12   enum opal_card_type { ADULT, STUDENT, CONCESSION };
```

struct vs. enum

Okay, but what about **enum** vs **#define**?

```
12    enum opal_card_type { ADULT, STUDENT, CONCESSION };
13
14    #define ADULT 0
15    #define STUDENT 1
16    #define CONCESSION 2
```

enum vs. #define

Now, let’s code up a coffee shop!