```c
// A program that says hello to the world

#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```c
// get_string and printf with incorrect placeholder

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string answer = get_string("What's your name? ");
    printf("hello, answer\n");
}
```

```c
// Conditionals, Boolean expressions, relational operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for integers
    int x = get_int("What's x? ");
    int y = get_int("What's y? ");

    // Compare integers
    if (x < y)
    {
        printf("x is less than y\n");
    }
}
```

```c
// Conditionals, Boolean expressions, relational operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for integers
    int x = get_int("What's x? ");
    int y = get_int("What's y? ");

    // Compare integers
    if (x < y)
    {
        printf("x is less than y\n");
    }
    else
    {
        printf("x is not less than y\n");
    }
}
```

```c
// Conditionals, Boolean expressions, relational operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for integers
    int x = get_int("What's x? ");
    int y = get_int("What's y? ");

    // Compare integers
    if (x < y)
    {
        printf("x is less than y\n");
    }
    else if (x > y)
    {
        printf("x is greater than y\n");
    }
    else if (x == y)
    {
        printf("x is equal to y\n");
    }
}
```

```c
// Logical operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user to agree
    char c = get_char("Do you agree? ");

    // Check whether agreed
    if (c == 'Y' || c == 'y')
    {
        printf("Agreed.\n");
    }
    else if (c == 'N' || c == 'n')
    {
        printf("Not agreed.\n");
    }
}
```

```c
// Opportunity for better design

#include <stdio.h>

int main(void)
{
    printf("meow\n");
    printf("meow\n");
    printf("meow\n");
}
```

```c
// Better design

#include <stdio.h>

int main(void)
{
    int i = 0;
    while (i < 3)
    {
        printf("meow\n");
        i++;
    }
}
```

```c
// Better design

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        printf("meow\n");
    }
}
```

```c
// Abstraction

#include <stdio.h>

void meow(void);

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        meow();
    }
}

// Meow once
void meow(void)
{
    printf("meow\n");
}
```

```c
// Abstraction with parameterization

#include <stdio.h>

void meow(int n);

int main(void)
{
    meow(3);
}

// Meow some number of times
void meow(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("meow\n");
    }
}
```

```c
// Prints a row of 4 question marks

#include <stdio.h>

int main(void)
{
    printf("????\n");
}
```

```c
// Prints a row of 4 question marks with a loop

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 4; i++)
    {
        printf("?");
    }
    printf("\n");
}
```

```c
// Prints a column of 3 bricks with a loop

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        printf("#\n");
    }
}
```

```c
// Prints a 3-by-3 grid of bricks with nested loops

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Prints a 3-by-3 grid of bricks with nested loops using a constant

#include <stdio.h>

int main(void)
{
    const int n = 3;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Prints an n-by-n grid of bricks with nested loops

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n = get_int("Size: ");

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Prints an n-by-n grid of bricks, re-prompting user for positive integer

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n = get_int("Size: ");
    while (n < 1)
    {
        n = get_int("Size: ");
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Prints an n-by-n grid of bricks, re-prompting user for positive integer

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Get size of grid
    int n;
    do
    {
        n = get_int("Size: ");
    }
    while (n < 1);

    // Print grid of bricks
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Prints an n-by-n grid of bricks, re-prompting user for positive integer

#include <cs50.h>
#include <stdio.h>

int get_size(void);
void print_grid(int n);

int main(void)
{
    int n = get_size();
    print_grid(n);
}

int get_size(void)
{
    int n;
    do
    {
        n = get_int("Size: ");
    }
    while (n < 1);
    return n;
}

void print_grid(int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

```c
// Addition with int, allows for overflow

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    int x = get_int("x: ");

    // Prompt user for y
    int y = get_int("y: ");

    // Perform addition
    printf("%i\n", x + y);
}
```

```c
// Addition with long

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Perform addition
    printf("%li\n", x + y);
}
```

```c
// Division with longs, demonstrating truncation

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Divide x by y
    printf("%li\n", x / y);
}
```

```c
// Division with longs, demonstrating truncation

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Divide x by y
    float z = x / y;
    printf("%f\n", z);
}
```

```c
// Division with longs, demonstrating type casting

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Divide x by y
    float z = (float) x / (float) y;
    printf("%f\n", z);
}
```

```c
// Division with longs, demonstrating floating-point imprecision

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Divide x by y
    float z = (float) x / (float) y;
    printf("%.20f\n", z);
}
```

```c
// Division with longs, demonstrating double

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    long x = get_long("x: ");

    // Prompt user for y
    long y = get_long("y: ");

    // Divide x by y
    double z = (double) x / (double) y;
    printf("%.20f\n", z);
}
```