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

#include <stdio.h>

int main(void)

printf("hello, world\n");
}
```

```
// 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");
}
```

```
1  // get_string and printf with %s
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8     string answer = get_string("What's your name? ");
9     printf("hello, %s\n", answer);
10 }
```

```
1
    // Conditional, Boolean expression, relational operator
 2
    #include <cs50.h>
3
    #include <stdio.h>
 6
    int main(void)
    {
        // Prompt user for integers
 8
        int x = get int("What's x? ");
 9
        int y = get_int("What's y? ");
10
11
12
        // Compare integers
13
        if (x < y)
14
        {
            printf("x is less than y\n");
15
16
17
   }
```

```
// Conditionals that are mutually exclusive
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 6
    int main(void)
    {
        // Prompt user for integers
 8
        int x = get int("What's x? ");
 9
        int y = get int("What's y? ");
10
11
12
        // Compare integers
        if (x < y)
13
14
        {
15
            printf("x is less than y\n");
16
17
        else
18
19
            printf("x is not less than y\n");
20
21
    }
```

```
// Conditionals that aren't mutually exclusive
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
    int main(void)
 6
    {
 8
        // Prompt user for integers
        int x = get int("What's x? ");
 9
        int y = get int("What's y? ");
10
11
12
        // Compare integers
13
        if (x < y)
14
        {
15
            printf("x is less than y\n");
16
17
        if (x > y)
18
19
            printf("x is greater than y\n");
20
        if (x == y)
21
22
23
            printf("x is equal to y\n");
24
        }
25
    }
```

```
// Conditional that isn't necessary
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
    int main(void)
 6
 8
        // Prompt user for integers
        int x = get int("What's x? ");
 9
        int y = get int("What's y? ");
10
11
12
        // Compare integers
13
        if (x < y)
14
        {
15
            printf("x is less than y\n");
16
17
        else if (x > y)
18
19
            printf("x is greater than y\n");
20
21
        else if (x == y)
22
23
            printf("x is equal to y\n");
24
        }
25
    }
```

```
1
    // Comparing against lowercase char
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 5
 6
    int main(void)
    {
        // Prompt user to agree
 8
        char c = get char("Do you agree? ");
 9
10
11
        // Check whether agreed
12
        if (c == 'y')
13
        {
14
            printf("Agreed.\n");
15
        else if (c == 'n')
16
17
18
            printf("Not agreed.\n");
19
        }
20
    }
```

```
1
    // Comparing against lowercase and uppercase char
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 5
 6
    int main(void)
 7
    {
 8
        // Prompt user to agree
        char c = get char("Do you agree? ");
 9
10
11
        // Check whether agreed
12
        if (c == 'y')
13
        {
14
             printf("Agreed.\n");
15
16
        else if (c == 'Y')
17
18
             printf("Agreed.\n");
19
        else if (c == 'n')
20
21
22
             printf("Not agreed.\n");
23
24
        else if (c == 'N')
25
            printf("Not agreed.\n");
26
27
        }
28
    }
```

```
1
    // Logical operators
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 5
 6
    int main(void)
 7
    {
        // Prompt user to agree
 8
        char c = get char("Do you agree? ");
 9
10
11
        // Check whether agreed
12
        if (c == 'Y' || c == 'y')
13
14
            printf("Agreed.\n");
15
        else if (c == 'N' || c == 'n')
16
17
18
            printf("Not agreed.\n");
19
        }
20
    }
```

```
1  // Opportunity for better design
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7     printf("meow\n");
8     printf("meow\n");
9     printf("meow\n");
10 }
```

```
// Better design
 1
 2
    #include <stdio.h>
    int main(void)
 6
         int i = 3;
while (i > 3)
 8
 9
10
              printf("meow\n");
11
             i--;
12
13
         }
    }
```

```
// Better design
 1
 2
    #include <stdio.h>
 4
 5
    int main(void)
 6
         int i = 3;
while (i > 3)
 7
 8
 9
10
              printf("%i\n", i);
11
             i--;
12
13
         }
    }
```

```
// Better design
 1
 2
     #include <stdio.h>
     int main(void)
 6
         int i = 0;
while (i < 3)</pre>
 7
 8
 9
              printf("meow\n");
10
11
              i++;
12
13
         }
    }
```

```
1  // Better design
2  
3  #include <stdio.h>
4  
5  int main(void)
6  {
7    for (int i = 0; i < 3; i++)
8    {
9       printf("meow\n");
10    }
11  }</pre>
```

```
// Abstraction
 1
    #include <stdio.h>
 3
    void meow(void);
    int main(void)
 8
 9
        for (int i = 0; i < 3; i++)
10
11
            meow();
12
13
    }
14
15
    // Meow once
16
    void meow(void)
17
        printf("meow\n");
18
19
```

```
// Abstraction with parameterization
 1
 2
    #include <stdio.h>
 3
    void meow(int n);
 6
    int main(void)
        meow(3);
 9
10
11
12
    // Meow some number of times
    void meow(int n)
13
14
        for (int i = 0; i < n; i++)</pre>
15
16
17
             printf("meow\n");
18
19
    }
```

```
// Addition with int
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
    int main(void)
 6
    {
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
12
        int y = get_int("y: ");
13
14
        // Perform addition
        printf("%i\n", x + y);
15
16
   }
```

```
1
    // Scope error
 2
    #include <cs50.h>
3
    #include <stdio.h>
 5
 6
    int add(void);
 7
 8
    int main(void)
9
    {
10
        // Prompt user for x
11
        int x = get_int("x: ");
12
        // Prompt user for y
13
14
        int y = get_int("y: ");
15
16
        // Perform addition
17
        int z = add();
18
        printf("%i\n", z);
19
    }
20
    int add(void)
21
22
    {
23
        return x + y;
24
    }
```

```
// Helper function with arguments and return value
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 5
 6
    int add(int a, int b);
 7
    int main(void)
 8
 9
    {
        // Prompt user for x
10
11
        int x = get_int("x: ");
12
        // Prompt user for y
13
        int y = get_int("y: ");
14
15
        // Perform addition
16
17
        int z = add(x, y);
18
        printf("%i\n", z);
19
    }
20
21
    int add(int a, int b)
22
    {
23
        int c = a + b;
24
        return c;
25
    }
```

```
1
    // Eliminates unnecessary variables
 2
    #include <cs50.h>
3
    #include <stdio.h>
 6
    int add(int a, int b);
 8
    int main(void)
9
    {
        // Prompt user for x
10
11
        int x = get_int("x: ");
12
        // Prompt user for y
13
        int y = get_int("y: ");
14
15
        // Perform addition
16
17
        printf("%i\n", add(x, y));
18
19
    int add(int a, int b)
20
21
22
        return a + b;
23
    }
```

```
// Addition with long
 1
 2
 3
    #include <cs50.h>
    #include <stdio.h>
    int main(void)
 6
 7
    {
 8
        // Prompt user for x
        long x = get_long("x: ");
 9
10
11
        // Prompt user for y
12
        long y = get long("y: ");
13
14
        // Perform addition
        printf("%li\n", x + y);
15
16
   }
```

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

#include <stdio.h>

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

```
1  // Prints a column of 3 bricks with a loop
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7    for (int i = 0; i < 3; i++)
8    {
9       printf("#\n");
10    }
11 }</pre>
```

```
// Prints a 3-by-3 grid of bricks with nested loops
 1
 2
3
    #include <stdio.h>
 5
    int main(void)
6
        for (int i = 0; i < 3; i++)
7
 8
            for (int j = 0; j < 3; j++)
 9
10
11
                printf("#");
12
            printf("\n");
13
14
        }
15 }
```

```
// Prints a 3-by-3 grid of bricks with nested loops using a constant
 1
 2
 3
    #include <stdio.h>
 4
 5
    int main(void)
 6
 7
        const int n = 3;
        for (int i = 0; i < n; i++)
 8
 9
10
            for (int j = 0; j < n; j++)
11
12
                printf("#");
13
14
            printf("\n");
15
        }
16
    }
```

```
// Prints an n-by-n grid of bricks with nested loops
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
    int main(void)
 6
    {
        int n = get_int("Size: ");
 8
 9
10
        for (int i = 0; i < n; i++)
11
12
            for (int j = 0; j < n; j++)
13
14
                printf("#");
15
            printf("\n");
16
17
        }
18
    }
```

```
// Prints an n-by-n grid of bricks, re-prompting user for positive integer
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 5
    int main(void)
 6
 7
    {
        int n = get_int("Size: ");
 8
        while (n < \overline{1})
 9
10
11
             n = get_int("Size: ");
12
        }
13
14
        for (int i = 0; i < n; i++)</pre>
15
16
             for (int j = 0; j < n; j++)
17
18
                 printf("#");
19
             printf("\n");
20
21
        }
22
    }
```

```
// Prints an n-by-n grid of bricks, re-prompting user for positive integer
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 6
    int main(void)
    {
        // Get size of grid
 8
        int n;
 9
10
        do
11
12
            n = get int("Size: ");
13
        while (n < 1);
14
15
        // Print grid of bricks
16
17
        for (int i = 0; i < n; i++)
18
        {
19
            for (int j = 0; j < n; j++)
20
21
                printf("#");
22
23
            printf("\n");
24
        }
25
    }
```

```
// Division with ints, demonstrating truncation
 1
2
3
    #include <cs50.h>
    #include <stdio.h>
    int main(void)
 6
7
    {
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
12
        int y = get_int("y: ");
13
14
        // Divide x by y
        printf("%i\n", x / y);
15
16
   }
```

```
// Uses %f
 1
 2
 3
    #include <cs50.h>
    #include <stdio.h>
 6
    int main(void)
 7
    {
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
        int y = get_int("y: ");
12
13
14
        // Divide x by y
15
        printf("%f\n", x / y);
16 }
```

```
// Uses float variable
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
    int main(void)
 6
 7
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
12
        int y = get_int("y: ");
13
14
        // Divide x by y
15
        float z = x / y;
16
        printf("%f\n", z);
17 }
```

```
1
    // Type casting
 2
 3
    #include <cs50.h>
    #include <stdio.h>
    int main(void)
 6
 7
    {
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
12
        int y = get_int("y: ");
13
14
        // Divide x by y
15
        float z = (float) x / (float) y;
16
        printf("%f\n", z);
17 }
```

```
// Floating-point imprecision
 1
 2
 3
    #include <cs50.h>
    #include <stdio.h>
    int main(void)
 6
 7
    {
 8
        // Prompt user for x
        int x = get_int("x: ");
 9
10
11
        // Prompt user for y
12
        int y = get int("y: ");
13
14
        // Divide x by y
15
        float z = (float) x / (float) y;
16
        printf("%.20f\n", z);
17 }
```

```
// Division with longs, demonstrating double
 1
 2
    #include <cs50.h>
 3
    #include <stdio.h>
 6
    int main(void)
 7
    {
        // Prompt user for x
 8
        long x = get long("x: ");
 9
10
11
        // Prompt user for y
12
        long y = get long("y: ");
13
14
        // Divide x by y
        double z = (double) \times / (double) y;
15
        printf("%.20f\n", z);
16
17 }
```