

---

```
1 // A program that says hello to the world
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("hello, world\n");
8 }
```

---

```
1 // get_string and printf with incorrect placeholder
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, answer\n");
10 }
```

```
1 // Conditionals, Boolean expressions, relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for integers
9     int x = get_int("What's x? ");
10    int y = get_int("What's y? ");
11
12    // Compare integers
13    if (x < y)
14    {
15        printf("x is less than y\n");
16    }
17 }
```

```
1 // Conditionals, Boolean expressions, relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for integers
9     int x = get_int("What's x? ");
10    int y = get_int("What's y? ");
11
12    // Compare integers
13    if (x < y)
14    {
15        printf("x is less than y\n");
16    }
17    else
18    {
19        printf("x is not less than y\n");
20    }
21 }
```

```
1 // Conditionals, Boolean expressions, relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for integers
9     int x = get_int("What's x? ");
10    int y = get_int("What's y? ");
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 // Logical operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user to agree
9     char c = get_char("Do you agree? ");
10
11     // Check whether agreed
12     if (c == 'Y' || c == 'y')
13     {
14         printf("Agreed.\n");
15     }
16     else if (c == 'N' || c == 'n')
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 }
```

---

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

```
1 // Abstraction
2
3 #include <stdio.h>
4
5 void meow(void);
6
7 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 {
18     printf("meow\n");
19 }
```

```
1 // Abstraction with parameterization
2
3 #include <stdio.h>
4
5 void meow(int n);
6
7 int main(void)
8 {
9     meow(3);
10 }
11
12 // Meow some number of times
13 void meow(int n)
14 {
15     for (int i = 0; i < n; i++)
16     {
17         printf("meow\n");
18     }
19 }
```

---

```
1 // Prints a row of 4 question marks
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("????\n");
8 }
```

---

```
1 // Prints a row of 4 question marks with a loop
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     for (int i = 0; i < 4; i++)
8     {
9         printf("?");
10    }
11    printf("\n");
12 }
```

---

```
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 }
```

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

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



```
1 // Prints an n-by-n grid of bricks with nested loops
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     int n = get_int("Size: ");
9
10    for (int i = 0; i < n; i++)
11    {
12        for (int j = 0; j < n; j++)
13        {
14            printf("#");
15        }
16        printf("\n");
17    }
18 }
```

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

```
1 // Prints an n-by-n grid of bricks, re-prompting user for positive integer
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get size of grid
9     int n;
10    do
11    {
12        n = get_int("Size: ");
13    }
14    while (n < 1);
15
16    // Print grid of bricks
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 }
```

```
1 // Prints an n-by-n grid of bricks, re-prompting user for positive integer
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int get_size(void);
7 void print_grid(int n);
8
9 int main(void)
10 {
11     int n = get_size();
12     print_grid(n);
13 }
14
15 int get_size(void)
16 {
17     int n;
18     do
19     {
20         n = get_int("Size: ");
21     }
22     while (n < 1);
23     return n;
24 }
25
26 void print_grid(int n)
27 {
28     for (int i = 0; i < n; i++)
29     {
30         for (int j = 0; j < n; j++)
31         {
32             printf("#");
33         }
34         printf("\n");
35     }
36 }
```

```
1 // Addition with int, allows for overflow
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     int x = get_int("x: ");
10
11     // Prompt user for y
12     int y = get_int("y: ");
13
14     // Perform addition
15     printf("%i\n", x + y);
16 }
```

```
1 // Addition with long
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Perform addition
15     printf("%li\n", x + y);
16 }
```

```
1 // Division with longs, demonstrating truncation
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Divide x by y
15     printf("%li\n", x / y);
16 }
```

```
1 // Division with longs, demonstrating truncation
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Divide x by y
15     float z = x / y;
16     printf("%f\n", z);
17 }
```



```
1 // Division with longs, demonstrating type casting
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Divide x by y
15     float z = (float) x / (float) y;
16     printf("%f\n", z);
17 }
```

```
1 // Division with longs, demonstrating floating-point imprecision
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Divide x by y
15     float z = (float) x / (float) y;
16     printf("%.20f\n", z);
17 }
```

```
1 // Division with longs, demonstrating double
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     long x = get_long("x: ");
10
11     // Prompt user for y
12     long y = get_long("y: ");
13
14     // Divide x by y
15     double z = (double) x / (double) y;
16     printf("%.20f\n", z);
17 }
```