Unit 10: Assertion

why?

Learning Objectives

- what if condition doesnt happen?
- isit for readability?

After this unit, students should:

- know what is an assertion
- be able to derive assertion statements in an algorithm that involves assignment and branches
- be able to note down assertions using Hoare's notation
- be able to apply De Morgan's laws

What is Assertion?

An assertion is a logical expression that must always be true for the program to be correct. We can write assertions either as part of the comment for the code or use the `assert()` macro in C. Let's look at what is assertion first. We will introduce the use of `assert()` later.

To get started, let's first look at the most trivial assertion:

```
1  long x = 1;
2  // { x == 1 }
```

just a comment

The line above initialize the variable `x` to be `1`. The next line, a comment, uses the curly braces `{` and `}` with a logical expression in between, to indicate that `x` must be equals to `1` after the assignment. We use the curly braces as a notation in CS1010, following C. A. R. Hoare's notation, but this is not part of any C standard.

The assertion above is kind of trivial and not very meaningful.

Let's revisit this snippet:

```
1  if (x > y) {
2    max = x;
3  } else {
4    max = y;
5  }
```

Let's consider the true block and the false block. Inside the true block, since `x > y`, we can assert that, well, `x > y`, and inside the false block, we have the negation, so `x <= y`.

```
1  if (x > y) {
2    // { x > y }
3    max = x;
4  } else {
5    // { x <= y }
6    max = y;
7  }
```

Let's now consider what happens after initializing `max` to either `x` or `y`.

```
1  if (x > y) {
2    // { x > y }
3    max = x;
4    // { max == x && max > y }
5  } else {
6    // { x <= y }
7    max = y;
8    // { max >= x && max == y }
9  }
```

The assertion on Line 4 consists of two parts: `max == x` which is the result of the assignment (the trivial assertion), but since inside this block, `x > y`, we must have `max > y` to be true as well.

Similarly, we can argue the assertion in Line 8 to be true.

What can we assert after we exit from the `if-else` block? We have either `max == x && max > y` or `max >= x && max == y`. This is exactly the property we are looking for in `max` when we set it to the maximum of `x` and `y`!

Let's look at another example:

```
1    if (score >= 8) {
2      cs1010_println_string("A");
3    } else {
4      if (score >= 5) {
5        cs1010_println_string("B");
6      } else {
7        if (score >= 3) {
8          cs1010_println_string("C");
9        } else {
10         cs1010_println_string("D");
11       }
12     }
13   }
```

Let's focus on the case of printing `C`. We should print `C` if `score` is less than 5 but is 3 or higher. Let's check if this is correct by finding out what we can assert wrt `score` just before printing `C`. We first add the assert condition to all the true blocks and the false block by negating the `if` condition.

```
1   if (score >= 8) {
2     cs1010_println_string("A");
3   } else {
4     if (score >= 5) {
5       // { score >= 5 }
6       cs1010_println_string("B");
7     } else {
8       // { score < 5 }
9       if (score >= 3) {
10        // { score < 5 && score >= 3 }
11        cs1010_println_string("C");
12      } else {
13        // { score < 5 && score < 3 }
14        cs1010_println_string("D");
15      }
16    }
17  }
```

We can see that, we are printing `C` when `score < 5 && score >= 3`, which is what we want.

Note that the last assert `score < 5 && score < 3` can be simplified to `score < 3`.

De Morgan's Law

To write an assertion for the false block, it is useful to know the De Morgan's law, which tells us how to negate some logical expression. Suppose we have two logical expressions `e1` and `e2`.

- `!(e1 && e2)` is the same as `(!e1) || (!e2)`
- `!(e1 || e2)` is the same as `(!e1) && (!e2)`

We have actually seen it in action. Recall the expression for Generation Z:

`(birth_year >= 1995) && (birth_year <= 2005)`.

To check for NOT Generation Z, we can write it as

`!((birth_year >= 1995) && (birth_year <= 2005))`,

which according to De Morgan's law, is the same as

`!(birth_year >= 1995) || !(birth_year <= 2005)`,

which is just

`(birth_year < 1995) || (birth_year > 2005)`,

exactly as we have written before!

Problem Set 10

Problem 10.1

Negate the following logical expression, then apply De Morgan's Law to simplify the resulting expression. Assume all variable names mentioned are boolean variables.

(a) `(x > 1) && (y != 10)`

(b) `!eating && drinking`

(C) `(has_cs2030 || has_cs2113) && has_cs2040c`

Problem 10.2

In the code below, replace `???` with the appropriate assertion. What will be printed?

```
long score = 4;
if (something) {
  score = 10;
} else {
  score = 0;
}
// { ??? }

if (score == 4) {
    score = 1;
} else {
    score += 10;
}
// { ??? }

if (score >= 10) {
    cs1010_println_string("ok");
} else {
    cs1010_println_string("failed");
}
```