CSC236H

Introduction to the Theory of Computation

Program Correctness

- **Program Correctness:** the program produces a **correct output** on every **acceptable input**.
  - Definitions of **acceptable inputs** and **correct output** depends on the problem that the program is designed to solve.

- Examining **correctness** of a given program:
  1. Running it on some set of test cases.
  2. Formally **prove** that the program is correct.

Program Correctness

- **Program Correctness:** the program produces a **correct output** on every **acceptable input**.

- **Program Specification:**

  - **Precondition**: an assertion which states what *must* be true before the program starts execution (acceptable inputs).

  - **Postcondition**: an assertion which states what must be true when the program terminate (correct output for the given input).

- A program is **correct with respect to a specification**, if whenever the precondition holds before the program starts execution, then
  1. the program terminates;
  2. after the termination, the postcondition holds.

Program Correctness

- A program is **correct with respect to a specification**, if whenever the precondition holds before the program starts execution, then
  1. the program terminates;
  2. after the termination, the postcondition holds.

- **Termination:**

- **Partial Correctness:**

Correctness of Recursive Programs – Example

- **Precondition:** $n \in \mathbb{N}$.
- **Postcondition:** Return n^2.

```
def Sq(n):
1    if n == 0:
2        return 0
3    else:
4        return Sq(n - 1) + 2n - 1
```

Correctness of Recursive Programs – Lessons Learns from the Example

- We do not prove termination and partial correctness separately, we encapsulate both ideas in a single predicate.
- The predicate must be defined over the **size** of the input.
- General form for the predicate:
  - $P(n)$: If precondition holds, and the program runs, and its input size is n, then the program halts and the postcondition holds after it halts.
  - Sanity check for identifying the correct predicate variable: The input for any recursive calls of the code is smaller than the input for the initial call of the code. So the input size must decrease in each recursive call.
- Must prove termination and the postcondition for all possible **paths**.

Correctness of Recursive Programs – Example

- **Precondition:** a and b are positive integers, and $a \geq b$.
- **Postcondition:** Returns the greatest common divisor of a and b.

```
def rec_gcd(a, b):
1    if a == 1 or b == 1:
2        return 1
3    else if a mod b == 0:
4        return b
5    else:
6        return rec_gcd(b, a mod b)
```