Background:
Problems:
specification of a procedure desired between input and output.
Procedures: tells how to behave step by step.

Example: Rem or Remainder
define Rem(n, m) = if n < m then return Rem(n-m, m)

Another interesting example:
define H(x) = if odd(x) then return ((3*x) + 1) / 2 else return x/2
define Steps(n) = if n==1 then return 0 else return 1+Steps(H(n))

Steps(1) == 0
Steps(2) == 1
Steps(3) == 5
..
..
Halting problem:
tekes a procedure of program p and input string I,
halts(p, I) = true, if P halts on input I
halts(p, I) = false, if p does not halt on input I
halts always returns true or false in a finite amount of time.

For you to do: describe an algorithm for solving the halting problem. Can you think of a possible way?

One suggestion: Check if the program enters to the same state twice. Still does not solve the problem since program may not enter the same state twice yet not halting

Halting problem is proved to be impossible to solve by Alan Turing. Can be proved by contradiction.

Rice's theorem: There is no program that can decide any non-trivial property of programs
Can programs help programmers?
Yes, but we cannot decide the followings -exactly-:

- Could the program encounter any type errors?
- Could the program ever crash?
- Will the program produce the correct output?

Programs can't decide these precisely, therefore, we must change the problem. We come up with similar problems that give us close answers.

Think of “Umbrella Today” website, which tells you if you need umbrella depending on your zip code.

Need umbrella	**It rains**	**No rain**
Yes	Good	OK
No	Bad	Good

Safety means: saying you need umbrella.

Compiler being conservative:
For some cases, it does not really know if you are going to a type error but it does not allow you compile
If read() **then return** 3 + 4 **else return** true + 4 **end**
If read evaluates to true all the time, you don’t have a type error. When it is false however, you have. Computer does not allow you to compile your code in this setting.

What it says	**Has error in run**	**No error in run**
Type error	Good	Ok
No error	Bad	Good

Sound, conservative type checking

We want our problem to be computable, we want answers from our type checker
Therefore, we have “sound” type checkers. If the program is wrong, it always says that the program is wrong. However, if your program is good, if may tell that “it might be not good”. So, we don’t have exact answers.

So, instead of exact answers like “yes” or “no”, you give answers like “maybe” and “I dont know”.
A workaround for halting problem.

ASTREE static analyzer
Terminator (Byron Cook et al., MSR, Cambridge): Checks if program fragments in C “might” loop forever. If it cannot certify that, it says “I don’t know”..

Conclusions: some tasks are impossible i.e., undecidable

Research in this area:

- Impossible problems give unlimited potential for advances.
- Working machine learning on halting problems might be interesting.

Program analysis is predicting statically safe approximations to the run-time behavior of programs
Static means before-run-time, while dynamic means at-run-time

To know more about:

- compiler optimization,
- program development tools,
- formal verification methods,
- Computer security
- Research in programming languages,

Take cop5021.

A type error is an attempt to use an operator outside its domain.
Matching number of arguments to a function, matching the format of the arguments
example: sqrt(-3, -5) the input should only one
-matching number of argument to a function
-matching format of argument

What types of guarantees does the typechecker give?
- function calls have parameters of expected type and number
-also for expressions
-arguments have expected types
-avoid undefined behaviour
-avoid lots of data
-avoid misinterpretation of data

One issue with typechecker: what if you have dynamic data (for example, read() function that reads any type of data)? How do we fit it into our typechecker?
Solution: you can have a special type (let's call it as 'dynamic'). Then, do dynamic type casts.

Example of cast: You can cast in C and Java. The nice thing about java is that even type casts happen in run-time, checks are still done. The dynamic aspects of types are checked statically.