```
int x=2, y=3;
return x+y;
```

and the empty list of parameters as the input, the output would need to be `true`. However, for the program

```
while (1);
return 23;
```

and the empty list of parameters the output would be `false`. Of course, we can write a program that will return `true` for all the programs that do terminate - this program merely needs to run its input program with the given inputs, and wait for the program to finish. If the input program terminates, then our analyser program will return `true`. But we have no way to tell if some program runs for a very long time whether it will eventually terminate (after, for example, 5 billion years), or it will never terminate. Therefore, this analyser program does not *decide* on the termination for each instance of a program and an input.

Very important observation here is that we do not claim that the analyser program that we want just currently doesn't exist. We claim something much stronger - that the analyser program *cannot* possibly exist on any current or future computer.

12.5 Examples of Undecidable Problems

In this section, we will give further examples of undecidable problems/languages. Each example that we give will be of a language that is partially decidable, but not totally decidable. For each such language L that we show to be partially but not totally decidable, the language $\overline{L}$ will give us another example of a language that is not even partially decidable.

12.5.1 Post's Correspondence Problem

Definition 12.3: Post's Correspondence Problem

Given two sequences of k strings, $(w_1, w_2, \ldots, w_k)$ and $(x_1, x_2, \ldots, x_k)$, find out if there exists a concatenation of corresponding strings from each list to form the same string.

We call the required concatenation a *solution* to the problem. We are, therefore, looking for an algorithm that will return "yes" if the solution exists, and "no" otherwise. For example, let $w_1 = 1, w_2 = 10111, w_3 = 10$ and $x_1 = 111, x_2 = 10, x_3 = 0$, then a solution in this case is

$$w_2w_1w_1w_3 = x_2x_1x_1x_3 = 101111110.$$

and our algorithm should return "yes". On the other hand, let $w_1 = 10, w_2 = 011, w_3 = 101$ and $x_1 = 101, x_2 = 11, x_3 = 011$. Then it is not hard to see that there is no solution. If there was a solution, it would have to start with w_1 and x_1, since these two are the only strings at the same position in their respective sequences where one is a prefix of the other. Now we have one string starting with 10 and the other starting with 101. Therefore, there is the last 1 in x_1 that is unmached in the concatenation of the strings w_1, w_2, w_3. Therfore, the next element in the concatenation has to be either w_1 and x_1 or w_3 and x_3. Assume it is w_1. Then on one side we have a concatenation $w_1w_1 = 1010$ and on the other $x_1x_1 = 101101$. These two concatenations do not match, therefore w_1w_1 (and x_1x_1) cannot be the start of a solution. Assume, instead, that w_3 and x_3 are the next strings in the concatenation. Then $w_1w_3 = 10101$ and $x_1x_3 = 101011$. There is a mismatch again. Therefore, for this input, there is no solution and our algorithm should return "no".

When seen as a language recognition problem, the Post's Correspondence Problem can be seen as a language

$$L_{\text{pcp}} = \{\langle (w_1, w_2, \ldots, w_k), (x_1, x_2, \ldots, x_k)\rangle \mid \text{there is a solution for } w_1, w_2, \ldots, w_k \text{ and } x_1, x_2, \ldots, x_k\},$$

and we want to determine the decidability of the language L_{pcp}. $\langle (w_1, w_2, \ldots, w_k), (x_1, x_2, \ldots, x_k)\rangle$ is some encoding of the sequences of strings $(w_1, w_2, \ldots, w_k)$ and $(x_1, x_2, \ldots, x_k)$ as binary strings.

It can be shown that if L_{pcp} is totally decidable, then so is L_{halt}, which we know is false. Therefore, L_{pcp} is not totally decidable. On the other hand, it is definitely partially decidable, since for any two input sequences, we can simply try all possible concatenations of the matching strings in some systematic way (e.g. first try all concatenations of one string, then all concatenations of two strings etc.), and we will eventually reach a solution, if one exists. But if the solution does not exist, we will loop forever.

12.5.2 Integer Solutions to Multivariate Polynomials

A *polynomial* of r variables can be defined using a recursive definition below.

Definition 12.4: Polynomial of r Variables With Integer Coefficients

Let $x_1, x_2, \dots, x_r$ be variables. A *polynomial of variables* $x_1, x_2, \dots, x_r$ *with integer coefficients* (or just *polynomial* for short) can be defined in the following way:

1. For any integer c, c is a polynomial.
2. For any positive integer k, x_i^k is a polynomial for every $1 \le i \le r$.
3. If P and Q are polynomials, so are $P + Q$, $P - Q$ and PQ.
4. Nothing else is is a polynomial.

An example of a polynomial with 1 variable with integer coefficients is $10x^6 + 24x^4 - 13x^3 + 28$, and an example of a polynomial with 2 variables with integer coefficients is $10x^2y^3 - 34xy^5 + 28xy - 30y - 11$.

Definition 12.5: Integer Solutions of Multivariate Polynomials Problem

Given a polynomial $P(x_1, x_2, \dots, x_r)$ of r variables with integer coefficients, determine whether P has integer roots. That is, whether or not some choice of integers $x_1, x_2, \dots, x_r$ leads to $P(x_1, x_2, \dots, x_r) = 0$.

Note that r is also a parameter of the problem. Some restricted variants of this problem, for example for $r = 1$ and $r = 2$ are decidable, but the general version of the problem above is undecidable. The problem is, however, partially decidable, as we can systematically try all the r-tuples of integers, starting from $x_1 = x_2 = \cdots = x_r = 0$ and calculate $P(x_1, x_2. \dots .x_r)$. If there exists an r-tuple for which $P(x_1, x_2, \dots, x_r) = 0$, then this algorithm will find it. However, if there does not exist such a tuple, the algorithm will keep on looping forever.